# COMPUTER SIMULATIONS OF REALISTIC MICROSTRUCTURES: IMPLICATIONS FOR SIMULATION-BASED MATERIALS DESIGN

A Dissertation
Presented to
The Academic Faculty

by

Harpreet Singh

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Materials Science and Engineering

Georgia Institute of Technology
April 2008

# COMPUTER SIMULATIONS OF REALISTIC MICROSTRUCTURES: IMPLICATIONS FOR SIMULATION-BASED MATERIALS DESIGN

Approved by:


Dr. Arun M. Gokhale, Advisor
School of Materials Science and Engineering
*Georgia Institute of Technology*

Dr. Hamid Garmestani
School of Materials Science and Engineering
*Georgia Institute of Technology*

Dr. Meilin Liu
School of Materials Science and Engineering
*Georgia Institute of Technology*

Dr. Karl I. Jacob
School of Polymer, Textile & Fiber Engineering
*Georgia Institute of Technology*

Dr. Steve W. Johnson
School of Materials Science and Engineering
*Georgia Institute of Technology*



Date Approved: December 11, 2007

*To my parents*

*Dhanwant Kaur and Balwinder Singh*

# ACKNOWLEDGEMENTS

I am very grateful to all the people who have supported me throughout my academic career. First, I would like to express my sincere gratitude to my thesis advisor, Dr. Arun Gokhale, for his invaluable guidance. I am truly thankful for his remarkable inspirations in problem solving and great patience. This work would not have been possible without his support and encouragement.

I wish to thank my committee members, Dr. Meilin Liu, Dr. Steve Johnson, Dr. Hamid Garmestani and Dr. Karl Jacob, for their time, effort and suggestions. I thank Mr. Gautam Patel for his expertise and help. I would also like to thank my colleagues Yuxiong Mao, Dr. Scott Lieberman, Arun Sreeranganathan and Dr. Soon-gee Li for their help and insights during the course of this research. I also wish to thank all my friends for their words of encouragement and for making my stay in Atlanta a memorable one.

This research has been supported through research grants from Air Force Office of Scientific Research (AFOSR grant numbers FA95550-05-1-0062 and F49620-01-1-0045), and Division of Materials Research, U.S. National Science Foundation (DMR-0404668). The financial support is gratefully acknowledged.

Above all, I wish to thank my family, specially my parents, Mr. Balwinder Singh and Mrs. Dhanwant Kaur. I would not be who I am today without my parents' unconditional love for me and sacrifices throughout their lives.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

The conventional route of materials development typically involves fabrication of numerous batches of specimens having a range of different microstructures generated via variations of process parameters and measurements of relevant properties of these microstructures to identify the combination of processing conditions that yield the material having desired properties. Clearly, such a trial and error based materials development methodology is expensive, time consuming, and inefficient. Consequently, it is of interest to explore alternate strategies that can lead to a decrease in the cost and time required for development of advanced materials such as composites. Availability of powerful and inexpensive computational power and progress in computational materials science permits advancement of modeling and simulations assisted materials design methodology that may require fewer experiments, and therefore, lower cost and time for materials development. The key facets of such a technology would be computational tools for (i) creating models to generate computer simulated realistic microstructures; (ii) capturing the process-microstructure relationship using these models; and (iii) implementation of simulated microstructures in the computational models for materials behavior. Therefore, development of a general and flexible methodology for simulations of realistic microstructures is crucial for the development of simulations based materials design and development technology. Accordingly, this research concerns development of such a methodology for simulations of realistic microstructures based on experimental quantitative stereological data on few microstructures that can capture relevant details of microstructural geometry (including spatial clustering and second phase particle

orientations) and its variations with process parameters in terms of a set of simulation parameters. The interpolation and extrapolation of the simulation parameters can then permit generation of atlas of "virtual" microstructures that covers the complete range of variations of processing conditions of interest. These simulated and "virtual" microstructures can then be used in the micromechanical models such as FEM to analyze their constitutive properties.

# CHAPTER 1

# PROBLEM FORMULATION AND RESEARCH OBJECTIVES

Mechanical and physical properties of materials significantly depend on the geometric attributes of their microstructure such as relative amounts of the constituent phases, their size and shape distributions and spatial arrangements [1-3]. The microstructure is in turn governed by the material chemistry and the processing conditions [1]. Therefore, processing-microstructure-properties relationships are of central importance in the materials design and development. The conventional route of materials development typically involves fabrication of numerous batches of specimens having a range of different microstructures generated via variations of process parameters, and measurements of relevant properties of these microstructures to identify the combination of processing conditions that yield the material having desired properties. Clearly, such a trial and error based materials development methodology is expensive, time consuming, and inefficient. Consequently, it is of interest to explore alternate strategies that can lead to a decrease in the cost and time required for development of advanced materials such as composites. Availability of powerful and inexpensive computational power and progress in computational materials science [3] permits advancement of modeling and simulations assisted materials design methodology that may require fewer experiments, and therefore, lower cost and time for materials development. The key facets of such a technology would be computational tools for (i) creating models to generate computer simulated realistic microstructures; (ii) capturing the process-microstructure relationship using these models; and (iii) implementation of

simulated microstructures in the computational models for materials behavior. Therefore, development of a general and flexible methodology for simulations of realistic microstructures is crucial for the development of simulations based materials design and development technology.

This research concerns development of a general, flexible, and efficient methodology for simulations of realistic microstructures based on experimental quantitative stereological data on few microstructures that can capture relevant details of microstructural geometry (including spatial clustering and second phase particle orientations) and its variations with process parameters in terms of a set of simulation parameters. The interpolation and extrapolation of the simulation parameters can then permit generation of atlas of "virtual" microstructures that covers the complete range of variations of processing conditions of interest. These simulated and "virtual" microstructures can then be used in the micromechanical models such as FEM to analyze their constitutive properties. Needless to mention, this is a complex and challenging problem whose complete solution will require contributions of numerous materials scientists over next decade or so. The work done through this research is a part of this new materials design methodology. Specifically, the major research objectives are as follows.

1. Development of a methodology for simulations of large windows of "realistic" two-phase microstructures that are statistically similar to the corresponding real microstructures with respect to complex realistic particle/feature shapes/morphologies; spatial clustering and correlations;

morphological orientation distributions; and size-shape-orientation distributions of the features.

2. Creation of "virtual" microstructures by correlating the simulation parameters with the processing parameters and then interpolating/extrapolating the curves.
3. Validation of the methodology through its applications to the microstructures of SiC reinforcement particles in a discontinuously reinforced Al-alloy matrix composites, and TiB whiskers in boron modified Ti-alloys

The research results provide significant for development of the technology for realistic microstructure-based design of materials.

# CHAPTER 2

# LITERATURE REVIEW AND BACKGROUND

## 2.1 Introduction

Simulations of microstructures at relevant length scales and implementation of such microstructural windows in the predictive models and simulations of materials behavior are important aspects of computational materials science. Clearly, such models and simulations of materials behavior can be reliable and useful only if they incorporate relevant *realistic* microstructural geometry. Accordingly, the focus of this research is on development of general and flexible methodology for simulations of *realistic* microstructures that can facilitate reliable predictions of materials behavior and simulations based efficient design and development of materials. The methodology is developed through its applications to the microstructures of discontinuously reinforced aluminum alloy (DRA) matrix composites containing SiC particles of complex shapes. Further validation of the methodology is carried out via simulations of the realistic microstructures of TiB whiskers in boron modified Ti-alloys. Accordingly, a brief background on these materials is given in the next two sections. The present research draws from existing stereological and image analysis techniques as well as from stochastic geometric microstructure simulation algorithms. The subsequent sections provide a background on the stereological, statistical and computational techniques employed in this research.

## 2.2 Discontinuously Reinforced Composites

The term "composite" broadly refers to a material system, which is composed of a discreet constituent (the reinforcement) distributed in a continuous phase called matrix [4]. Depending on the nature of the matrix, the composites are grouped into polymer matrix, metal matrix, and ceramic matrix composites. These are further classified on the basis of the geometry of the reinforcement as continuous and discontinuous fibers/particles reinforced composites. The present research concerns discontinuously reinforced composites and alloys. There are various parameters that characterize these tailored materials including the mechanical and physical properties of their constituents, properties of the interfaces among the constituent phases, and the geometric microstructural attributes such as the amount, size-shape-orientation distribution, and spatial distribution of the reinforcement phase. These parameters depend on the processing conditions, and they in turn affect the properties and performance of the composites. The following subsections give a brief description of the processing, microstructure, properties, and applications of the two metal matrix based materials of interest in the present work, namely, discontinuously-reinforced aluminum (DRA) composites containing SiC particles and boron modified Ti-alloys containing TiB whiskers.

### 2.2.1 DRA Composites

Discontinuously reinforced aluminum (DRA) composites belong to a special class of metal-matrix composites where the aluminum alloy matrix is reinforced with high strength constituents such as silicon carbide, aluminum oxide, or boron carbide in

particulate forms. The DRA composites having SiC particulate reinforcement have high strength and excellent interface characteristics. These DRA composites exhibit outstanding mechanical properties including high elastic moduli, yield strength, and wear resistance. Since the density of DRA composites is similar to those of conventional aluminum alloys, these composites have better specific properties than corresponding unreinforced aluminum alloys [4]. Figure 2.1 compares important mechanical properties of the DRA composites with those of other materials. The DRA composites can be processed using primary conventional metallurgical techniques such as casting and powder processing, and secondary deformation processing techniques such as extrusion, forging, and rolling. The important processing techniques are described in the following sub-section.

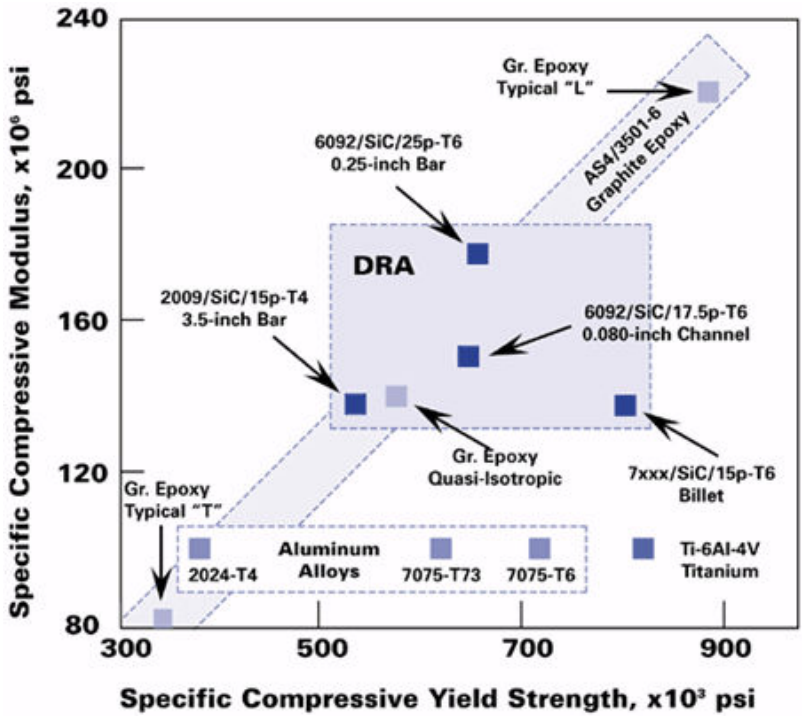



Figure 2.1: Comparison of properties for different materials [5]

### 2.2.2 Processing of DRA Composites

There are various processing routes available to manufacture aluminum matrix composites. These can be divided into two categories, namely, solid state and liquid state processes. The choice of a particular route is based upon the type and amount of reinforcement and the degree of microstructural homogeneity desired.

#### *2.2.2.1 Solid state processing*

Solid state processing techniques for metal matrix composites include the powder metallurgy (P/M) route and physical vapor deposition technique. Powder blending and consolidation (P/M processing) is one of the most common industrial method used to produce composites having aluminum matrix [6]. Powders of the alloy and the reinforcement are first blended and that is followed by cold compaction, canning, degassing, and high temperature consolidation steps such as hot isostatic pressing or extrusion. Figure 2.2 shows a schematic of a typical P/M processing route. The materials used in this research were manufactured using such processing techniques.

There are various process parameters that govern the final microstructure of the DRA composites produced via P/M processing route. These include size-shape distributions of initial powders, compaction pressure, extrusion temperature, extrusion ratio, and relative amounts of the constituent powders. One of the important processing parameter in the production of the DRA composites is the particle size ratio (PSR), which is defined as the ratio of the mean size of the matrix powder particles to the mean size of the reinforcement particles. It is known that PSR is an important factor that affects the spatial homogeneity of the reinforcement phase distribution in the composites

manufactured via powder metallurgy route [1]. Increasing the PSR leads to a reduction in the combined surface area of the matrix alloy particles and as this area becomes insufficient for a uniform arrangement of reinforcement particles, clusters of second phase particles are formed in-between the larger matrix particles. Figure 2.3 shows a schematic depicting the microstructures arising from different PSR values of the constituent powders. In the diagram, the light circles represent the matrix particles and the dark circles represent the reinforcement particles. Now, if the matrix particle size is increased keeping the reinforcement particles of the same size, the microstructure with the higher value of PSR tend to be more clustered, as depicted in the Figures 2.3a and 2.3b. Numerous other processing parameters affect microstructure of the DRA composites; these parameters include compaction pressure, extrusion temperature, and extrusion ratio. An increase in the compaction pressure causes a reduction in the resulting porosity of the composites [7]. As for the extrusion parameters, the increase in extrusion temperature increases the anisotropy in the distribution of the second phase particles and particle rich clusters, while reducing the overall porosity in the microstructure [8], on the other hand an increase in extrusion ratio causes more uniform distribution of reinforcement particles but increases the microstructural anisotropy [9].

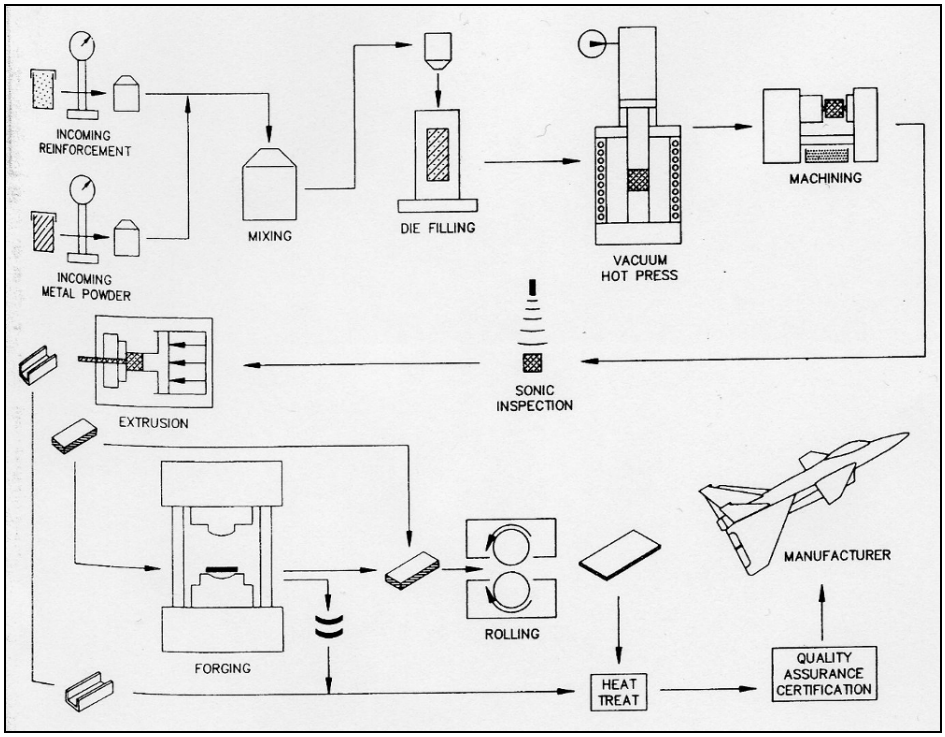
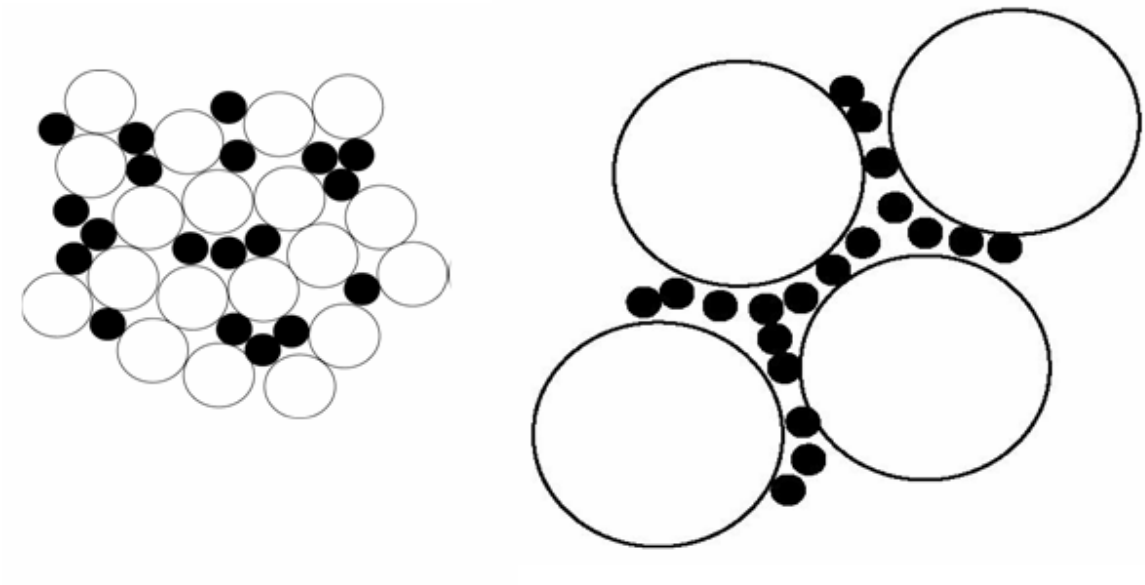



Figure 2.2: Schematic for PM processing [6]



(a) Low PSR (b) High PSR

Figure 2.3: Schematic comparing the microstructures with different PSR values

Another important solid state processing technique for producing metal matrix composites is physical vapor deposition (PVD). This process is mainly used for fiber-reinforced composites. First, the vapor of the metal to be deposited is produced by directing a high power electron beam onto the end of a solid bar feedstock. Fibers are passed through the region having high vapor pressure; the vapor condensation leads to formation of a relatively thick coating of the metal on the fibers. Typical deposition rates are on the order of 5-10 μm per minute. Coated fibers are then assembled together in an array that is consolidated in a hot press or by hot isostatic pressing (HIP) operation. Composites with uniform distribution of second phase and volume fractions up to 80% can be produced by this technique. This process can also be used to produce particle-reinforced composites.

*2.2.2.2 Liquid state processing*

The major methods under this category are stir casting, infiltration process and spray deposition [4]. Stir casting involves incorporation of ceramic particulates into liquid aluminum melt and subsequent solidification of the composite. An important consideration in this process is wetting between the particulate reinforcement and the liquid aluminum alloy melt as the poor wetting can lead to weak interfaces between the reinforcement and the matrix phases that can give rise to damage nucleation due to debonding. In the cast composites, particle agglomeration and inhomogeneity in the reinforcement distribution can occur due to interactions between suspended ceramic particles and moving solid-liquid interface during solidification, which can adversely

affect their fracture sensitive properties [4]. Stir casting process can produce composites with volume fraction up to 30%.

In the infiltration process, liquid aluminum alloy is injected/infiltrated into the interstices of the porous pre-forms of continuous particle/whisker/fiber to produce the composite [4]. Pressure or vacuum can also be used to infiltrate the liquid metal in the pre-form depending upon the nature of reinforcement and its volume fraction. Composites having volume fraction in the range of 10 to 70% can be produced using the infiltration technique.

Spray deposition involves the passing of reinforcement particles/whiskers through a droplet stream of aluminum [4]. The spray can either be produced from a molten bath (Osprey process) or by continuous feeding of cold metal into a zone of rapid heat injection (thermal spray process). The spray deposition technique invariably leads formation of about 5 to 10% porosity (damage), and therefore, subsequent secondary processing is required for reduction of porosity [4].

### 2.2.3 Microstructure of Al/$SiC_p$ Composites

In the present research, the microstructure simulation technique is developed via its application to the microstructures of SiC particles in the DRA composites. The microstructure of these composites consists of $SiC_P$ particles dispersed in an Al-alloy matrix. The spatial distribution of the particles can be almost uniform random (homogenous) or clustered depending on the process parameters such as PSR. In the extruded composites, the clusters (bands) of SiC rich regions are aligned along the extrusion direction. The extent of such anisotropy (banding) and the size and shape of the

bands depend on the process parameters such as extrusion ratio and extrusion temperature. These powder-processed composites also contain some porosity particularly in the SiC particle rich regions. Figure 2.4 shows an example of one such microstructure having clusters of SiC particles in the extrusion direction and presence of porosity in the particle clusters.

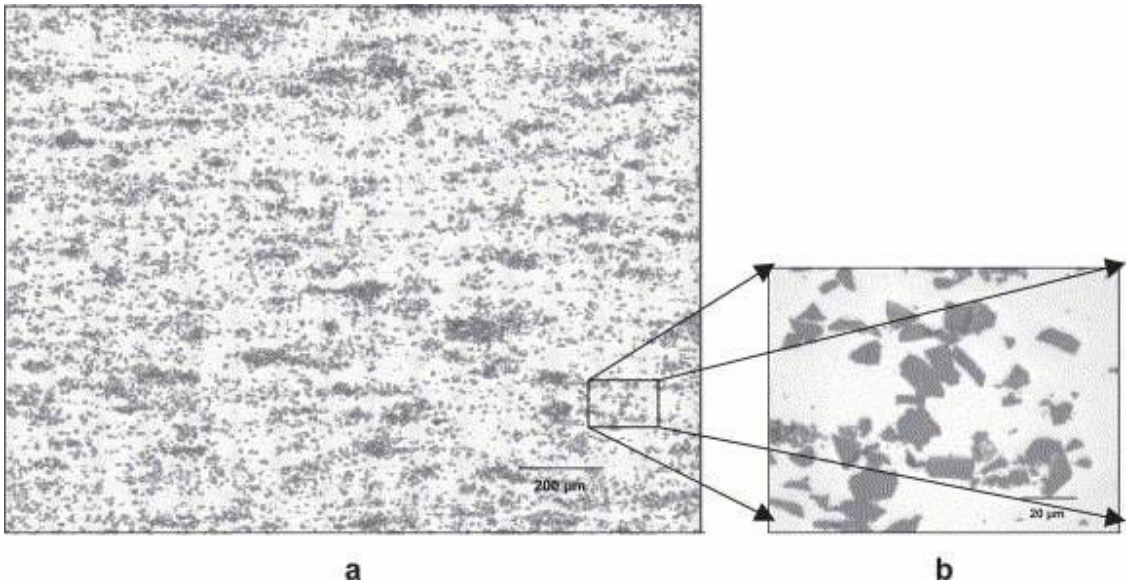



Figure 2.4: (a) Micrograph showing microstructure of Al/SiCp composite, (b) high resolution image depicting the complex shapes of SiC particles.

2.2.4 Properties and Applications of DRA Composites

DRA composites are primarily being looked upon as the replacement for the existing monolithic materials for structural applications. These composites have been successively used as the fan exit guide vane (FEGV) in the gas turbine engine (figure 2.7). Although, the initial major uses of the DRA composites were restricted to automotive and aerospace [10] applications (figure 2.5), improved tailored properties

coupled with economical and environmental benefits have extended their use in electrical and functional applications (figure 2.6) [11]. Other key application that utilizes the physical properties of particulate reinforced composites is in thermal management components for electronic packaging systems. The lightweight and high strength combination of DRA composites has also made them a useful material for recreational products.

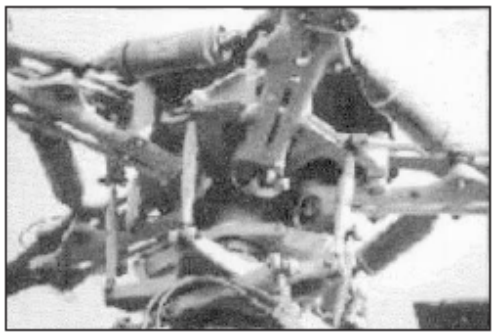


Figure 2.5: Aluminum matrix composite used in blade sleeves used in helicopters [4]

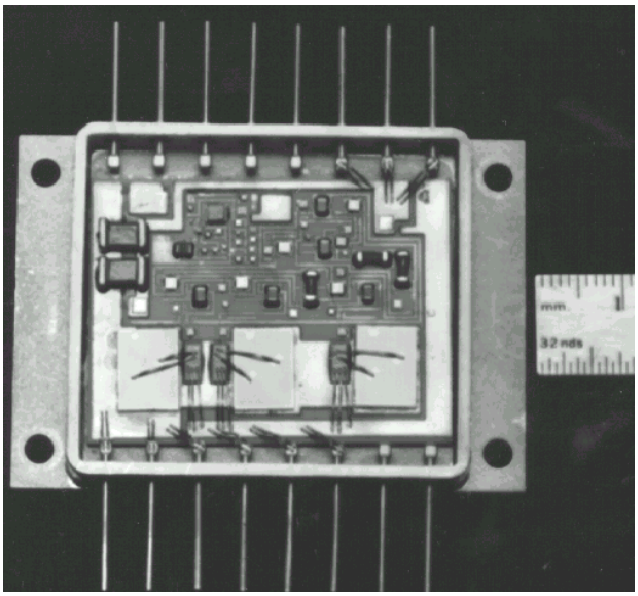


Figure 2.6: DRA electronic package for a remote power controller [11]

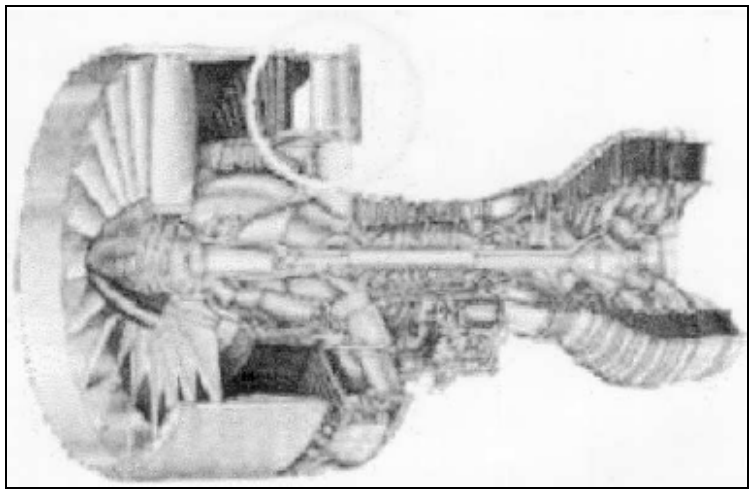


Figure 2.7: FEGV component used in gas turbine engine [4]

Even though particulate composites have been used in various applications, their poor fracture toughness and low fracture strain have limited their use in critical structural applications. The low fracture toughness of DRA the composites can be linked to the brittle reinforcement phase, which controls the fracture mechanisms [12-16]. Research done in this area suggests that the low fracture toughness of these composites can be attributed to two main problems. First, depending on the temperature, environment and other parameters, there is a possibility that the reaction between Al and SiC produces Al4C3, which weakens the interface that can lead to debonding damage initiation. The other reason is the presence of clusters of reinforcement particles, elongated in the extrusion direction. Clusters of particles are usually accompanied by the presence of porosity, as shown in figure 2.4. These voids assist in matrix cracking which can lead to failure. Studies done on the fracture mechanisms in the DRA composites show that even though the overall crack path is random; damage accumulation is often concentrated in

the regions of spatially clustered reinforcement particles. The presence of particle-porosity clusters also affects the wear resistance and friction behavior of the composites. This property is especially important since DRA composites find extensive use in the automotive engine applications such as cylinder blocks, pistons and piston insert rings, which demand high wear resistance [17]. Although, some attempts have been made to model the effects of spatial clustering on reinforcement particles on the fracture and wear properties of the composites [18], the success is very limited due to lack of practical techniques for characterization and mathematical representation of the spatial clustering of the reinforcement particles. Accordingly, one objective of the present research is to develop stereology and image analysis based flexible and general techniques for characterization and mathematical representation of spatial clustering and heterogeneity in these microstructures.

In summary, the mechanical and physical properties of the DRA composites depend on (1) volume fraction, (2) mean size, (3) morphologies/shapes, (4) spatial clustering, and (5) orientations/anisotropy of the reinforcement particles as well as on the constitutive behavior of the matrix that can be altered via heat treatment. The microstructural attributes are in turn governed by the process parameters such as (1) size-shape distribution of the initial constituent powder particles, (2) particle size ratio, and (3) powder compaction, extrusion, and heat treatment process conditions. Therefore, there are multi-parameter multi-variate processing-microstructure-properties relationships in these composites.

### 2.2.5 Titanium Alloys and Composites

Due to its high strength-to-weight ratio and excellent corrosion resistance, Ti – alloys are attractive for aerospace and other structural applications [19, 20]. Nonetheless, pure Ti and its alloys have relatively low modulus and wear resistance, which has motivated the research on the development of boron modified Ti-alloys and their composites [21]. These research efforts during the last decade or so have led to the development of discontinuously reinforced titanium matrix MMCs (also known as DRTi composites), and boron-modified Ti-alloys containing titanium boride whiskers. Extensive studies done on these materials have shown that important properties of conventional titanium alloys including strength, stiffness, wear resistance, and microstructural stability can be significantly improved by adding even relatively small additions of boron [22, 23]. A wide variety of processing methods are available for production of boron modified Ti-alloys and composites, and each of these processing routes permit wide variations in numerous processing conditions [24-31]. Therefore, this class of materials is ideally suited for applications of simulations based materials design. Consequently, in this study, research has been also conducted on simulations of microstructures of boron modified Ti-alloys to develop and validate the simulation methodology. Accordingly, a brief overview of processing, microstructure, properties, and applications of these alloys and composites is given below.

### 2.2.6 Processing of Boron Modified Titanium Alloys and Composites

Boron modified Ti-alloys and their composites can be produced using standard powder metallurgy (P/M) techniques, including mechanical alloying, blended elemental

P/M, hot isostatic pressing (HIP), direct powder extrusion, functionally graded material (FGM), laser cladding, reaction sintering, and self-propagating high-temperature synthesis (SHS). Rapid solidification techniques, such as gas atomization and ribbon melt spinning have also produced Ti-B alloys and composites. Conventional ingot metallurgy techniques, including vacuum arc remelting, induction skull melting, and vacuum induction melting can be also used for these materials. In addition, secondary hot forming techniques, including extrusion, swaging, rolling, and forging, can be used in the Ti-B material production process [32].

Among the numerous processing routes mentioned above, powder metallurgy (P/M) using pre-alloyed (PA) Ti-B powder is of significant current interest. Pre-alloying is a rapid solidification process where an alloy melt is rapidly solidified into an alloy powder by inert gas atomization. The PA powder can then be processed using conventional powder metallurgy processes, including outgassing to remove any volatile impurities and compaction by techniques such as hot isostatic pressing (HIP) to produce near-net shape products or billet preforms. These billet preforms can then be subjected to thermo-mechanical processing (TMP) methods such as forging, rolling, or extrusion to manufacture wrought products. Note the each of these processing steps and associated process parameters affect the microstructure attributes (for example, size, shape, and orientations of TiB whiskers their volume fraction, etc), and consequently, the mechanical properties of these materials. An important reason for the development of realistic microstructure simulations methodology is to capture these complex processing-microstructure relationships.

One of the interesting characteristic of Ti-B alloy and composite processing is realignment of the TiB whisker reinforcement during deformation processing steps such as extrusion and forging. Schuh and Dunand observed gradual alignment of TiB whiskers during tensile deformation by transformation and superplasticity in the range of 840 and 1030°C [33]. The TiB whiskers align along the external loading axis, showing no evidence of whisker fracture or interfacial debonding during reorientation. Tamirisakandala, Vedam, and Bhat published processing maps (strain rate versus temperature), for hot working of Ti-B composites [34]. Other processing techniques such as, hot extrusion, forging and swaging have also been shown to align the TiB whiskers. These observations present an interesting possibility of designing the microstructural anisotropy to optimize the mechanical properties anisotropy best suited for a given component via appropriate design of the deformation processing route. Realistic microstructure simulations can provide useful input to arrive at optimum microstructural geometry and process conditions for such applications.

### 2.2.7 Microstructure of Boron Modified Titanium Alloys and Composites

Figure 2.8 shows the Ti-B binary phase diagram. Boron is essentially insoluble in titanium, leading to a stable TiB intermetallic phase, which forms in situ during the eutectic reaction. Note that small amounts of boron in titanium will form a relatively high volume fraction of TiB. It has been shown that TiB is stable only in a titanium rich matrix, due to the thermodynamics of the reactions forming TiB from Ti and $TiB_2$. Ti-B materials containing less than the eutectic percentage of boron (approximately 1.55 wt% B for boron containing Ti-64 alloys) are called boron-modified titanium alloys, whereas those with a higher percentage of boron are referred to as Ti-B composites[32].

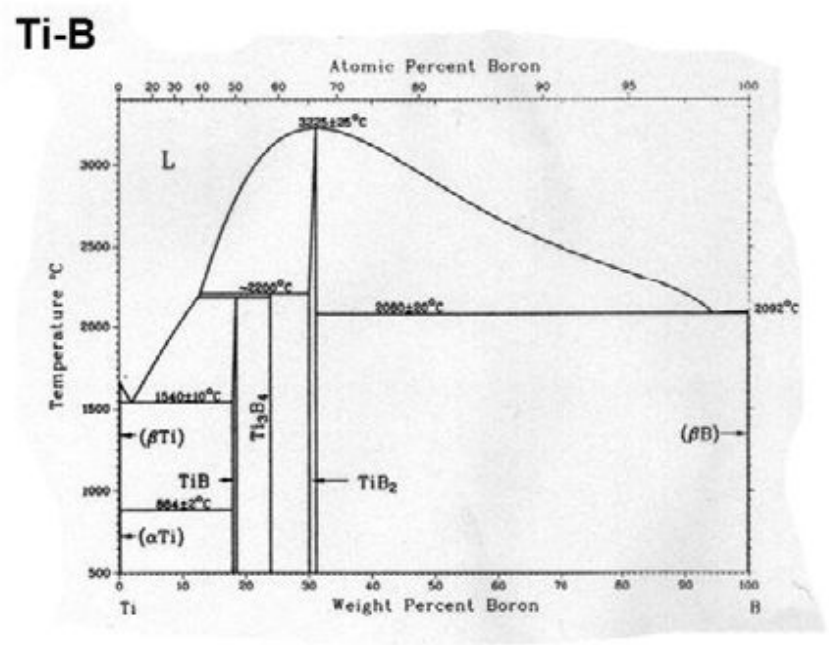



Figure 2.8: Ti-B binary alloy phase diagram [35].

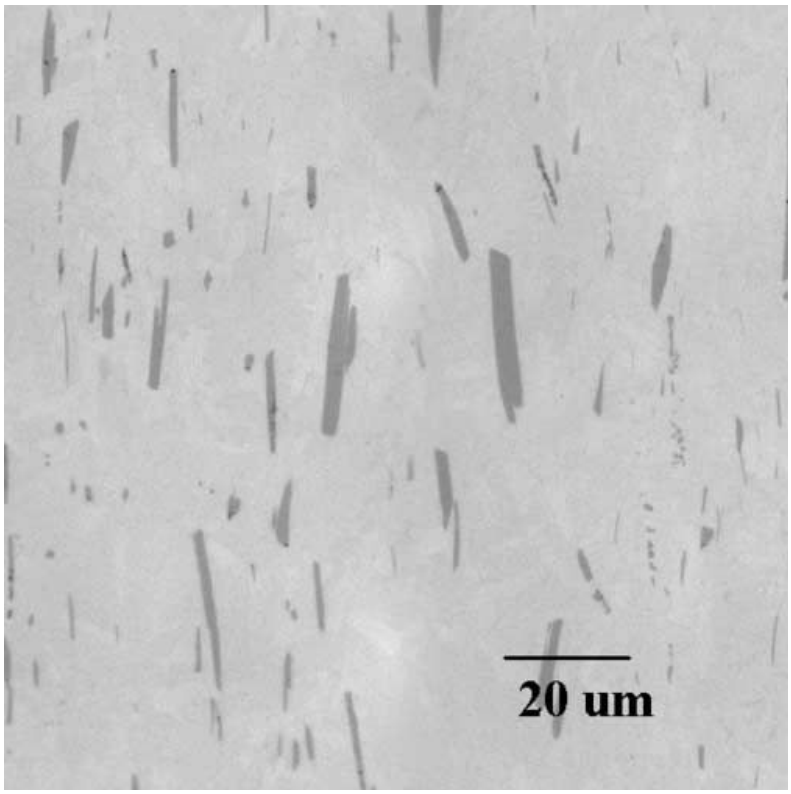



Figure 2.9: Microstructure of boron modified titanium alloy.

The microstructure of boron modified titanium alloys and composites typically consist of TiB particles/whiskers, uniformly distributed throughout a fine-grained equiaxed matrix. Figure 2.9 shows a micrograph depicting the microstructure of boron modified titanium alloy observed in a metallographic plane parallel to the extrusion direction. Studies have shown no evidence of a chemical reaction zone at the interface between the titanium matrix and the TiB phase and the interface is known to be nearly perfect such that it is atomically flat, leading to formation of a coherent phase boundary that is in thermodynamic equilibrium. The microstructure of the matrix depends on the alloy composition. In the boron modified Ti-6Al-4V alloy (which is of interest in the present work) and its composites, the matrix microstructure consists of fine equiaxed grains of alpha and beta Ti-rich phases. It has been shown that the TiB whiskers effectively pin the grain boundaries of the titanium alloy matrix so that a fine grain structure is retained even well above the β-transus and after cooling back into the α+β phase field [28, 36]. In the present work, the focus is on the microstructures of TiB whiskers rather than the matrix phases.

### 2.2.8 Properties and Applications of Boron Modified Ti-Alloys and Composites

Recall that the TiB phase forms in situ during the eutectic reaction $liquid \rightarrow \beta^{Ti} + TiB$ that occurs during the processing of boron modified Ti-alloys and composites, and the TiB is thermodynamically stable only in a Ti-rich matrix. Therefore, the mechanical properties of TiB whiskers (such as Young's modulus) cannot be determined using conventional macro-scale mechanical tests. Consequently, several studies have attempted to determine the mechanical properties the TiB phase using theoretical approaches [37].

Gorsse and Miracle observed a significant strengthening effect of TiB reinforcement in Ti-6Al-4V-TiB composites [38, 39]. They determined the elastic modulus of compacted 20 volume percent TiB MMC, E = 161 GPa; extruded 20% TiB MMC, E = 168 GPa; and extruded 40% TiB MMC, E = 199 GPa, which approaches the elastic modulus of standard steel alloys. The elastic modulus of unreinforced Ti-6Al-4V alloy is approximately 110 GPa. Constantinides, Ravi Chandran, Ulm, and Van Vliet developed a grid indentation analysis technique in an attempt to determine the elastic modulus of TiB in Ti-B composites [37]. They used massive array nanoindentation on reaction hot sintered Ti-B composite materials to generate statistical average mechanical properties. Their methods estimated the elastic modulus of TiB whiskers to be 406 GPa.

Alman and Hawk studied the wear resistance of titanium matrix composites, and found that adding $TiB_2$ to the titanium matrix, forming in situ TiB and $TiB_2$ phases, was effective in improving wear resistance over the unreinforced alloy [23]. They found that the composite possessed significantly lower wear coefficients than the unreinforced titanium alloy, with TiB-reinforced titanium composite more wear resistant than TiC-reinforced titanium composite which was also examined. They observed that strong interfacial bonding between the titanium matrix and the TiB whisker reinforcement prevents preferential pullout of either of the phases during abrasion.

The high strength-to-weight ratio, excellent corrosion resistance, and improved high temperature properties compared to the unreinforced alloy have made Ti-B modified alloys and composites an increasingly viable choice for applications within the automotive industry. Analysis done by Froes, Friedrich, Kiese, and Bergoint for the potential use of titanium alloy and composites versus steel in automobiles showed that

the main obstacle to increased titanium use versus steel is cost. Currently, a titanium-based part will cost more than 10 times that of a comparable steel part, and this needs to be reduced to approximately twice that of steel in order to see cost-performance benefits to justify a material switch. Their study noted the use of Ti-B composite connecting rods in the Volkswagen CCO diesel concept vehicle, where the reduced weight of the titanium composite part leads to significantly reduced noise, vibration, and harshness (NVH), and quieter engine performance in the vehicle that has a fuel economy of 250 miles per gallon (mpg).

Although relatively little of the research has been published, titanium matrix alloys and composites have been developed for the aerospace industry over the last half-century. Ti-B composite was used in the nozzle actuator piston rod for the Pratt & Whitney F-119 engine developed in 1995 for the F-22 fighter jet, and in the nozzle link for the General Electric F-110 engine developed in 1999 for the F-16 fighter jet [40]. In the industry, Ti-B modified alloys and composites are currently being considered as potential candidates for aerospace structural components requiring high strength, high stiffness, and low density; to replace high-strength steels due to Ti-B materials' superior wear and corrosion resistance; and for elevated temperature applications where standard unreinforced titanium alloys are unsuitable [41]. For fracture-critical aerospace applications, a minimum of 7% tensile elongation to failure is required by structural designers [42].

Titanium alloys and composites are now also being commercially used for the recreational applications. Dynamet Technology has developed a powder metallurgy cold and hot isostatic pressing (CHIP) process to produce a range of products for sporting

goods [43-45]. Marketed as CermeTi®, they produce Ti-6Al-4V-based composites with either TiB or TiC reinforcement. They have produced golf driver inserts, as well as knife blades and hockey skate blades, where the lighter weight and improved corrosion resistance over a traditional steel part are particularly useful.

Other applications being considered for the use of titanium alloys and composites with boron include certain biomedical applications. Research has investigated using titanium modified alloys and composites for load-bearing biomedical implants, such as femoral ball and lumbar disc replacements [34]. In addition to its corrosion resistance, high strength-to-weight ratio, biocompatibility, and osseo-integration, titanium alloys and composites do not obscure soft-tissue magnetic resonance imaging (MRI), unlike stainless steel and cobalt-chrome alloys. Other biomedical application for the Ti-B materials under study is their suitability for near-net shape dental castings used in dental applications such as implants and restorative castings [46].

## 2.3 Stereology and Image Analysis

A three-dimensional material microstructure can in general be defined as an arrangement of lines (e.g. dislocations), surfaces (e.g. interfaces) and volumes (e.g. particles). Any microstructure contains some or all of these geometric features. The spatial arrangement of the geometric features and their morphological orientations may or may not be uniform-random, and the geometrical shapes/morphologies of the microstructural features are often irregular and complex. As most of the materials are opaque, the microstructural observations are generally carried out on the two-dimensional (2D) metallographic sections through 3D microstructural domains of interest. The

microstructure observed in a metallographic section consists of intersections of the features in the 3D microstructure with the sectioning plane. Stereological and statistical descriptors can be useful in sampling the 3D microstructure using 2D metallographic planes.

Detailed mathematical/statistical representations of microstrucutral geometry are essential for simulations of realistic microstructures. These experimental data can be obtained using a combination of sterology, digital image processing, and three dimensional microstructure reconstruction techniques. The basic statistical descriptors that are used to characterize three-dimensional geometric parameters and three-dimensional microstructure reconstruction techniques are described in the following sub-sections.

### 2.3.1 Statistical Descriptors of Microstructure

There are numerous contributions in the literature that deal with the statistics of spatial point patterns and quantitative descriptors that reflect various attributes of the spatial arrangement of ensembles of features. Nearest neighbor distributions, radial distribution function, *n*-point probability function and lineal path correlation functions are some such examples. These descriptors are useful in describing spatial patterns like randomness, clustering, repulsion and short and long range interactions. The following sub-sections describe in detail the functions employed in this research namely, *n*-point probability and lineal path correlation functions.

*2.3.1.1 N-point Probability Functions*

There are numerous ways quantitatively to describe and mathematically represent a microstructure [47]. One class of such descriptors is *n*-point correlation functions. Consider a thought experiment involving placement of a polyhedron having *n* vertices at random locations in a three-dimensional (3D) microstructure of interest containing two phases, namely phase-1 (which may be particles, voids, inclusions, etc.) and phase-2 (matrix). The probability that all *n* vertices of the polyhedron are contained in phase-1 is an *n*-point correlation function that varies with the size, shape, and orientation of the polyhedron in the 3D space of the microstructure [48]. One can similarly define another *n*-point correlation function that represents the probability that all the *n* vertices of the polyhedron are in phase-2, and so on. Thus, one can formulate one-, two-, three-, … *n*-point correlation functions representing the corresponding probabilities. These statistical correlation functions implicitly contain information concerning the first-order microstructural parameters such as volume fractions, number densities, and size distributions, as well as spatial arrangements and morphological anisotropy of the features in 3D microstructure, and therefore they are useful for mathematical representation of microstructures. Statistical mechanics theories link correlation functions of a heterogeneous material microstructure to properties such as elastic constants and thermal conductivity [49-51].

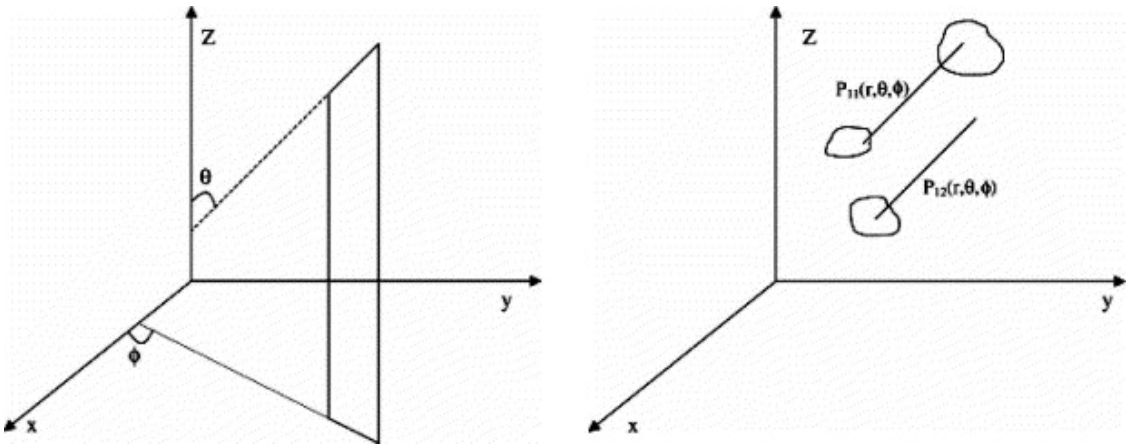



Figure 2.10: Schematic showing the relationship between $r$, $\theta$, and $\varphi$ for two-point correlation function $P_{11}(r, \theta, \varphi)$ and $P_{12}(r, \theta, \varphi)$ with $z$ as the extrusion axis.

The lowest order correlation function is the one-point correlation function, which is the probability that a randomly placed point in a 3D microstructure is contained in a given phase, and that is precisely equal to the volume fraction of that phase. For a two-phase microstructure, a two-point correlation function $P_{ij}(r, \theta, \varphi)$ is the probability that a straight line of length $r$ and angular orientation $(\theta, \varphi)$ randomly placed in a 3D microstructure is such that its first end is in the phase $i$ (where $i = 1$ or 2) and the second end is contained in the phase $j$ (where $j = 1$ or 2). Figure 2.10 shows a schematic of the relationship between $r$, $\theta$, and $\varphi$. Note that the probability associated with a two-point correlation function only concerns the events at the end points of the line. For a two-phase microstructure, there are four possible two-point correlation functions, namely $P_{11}(r, \theta, \varphi)$, $P_{22}(r, \theta, \varphi)$, $P_{12}(r, \theta, \varphi)$, and $P_{21}(r, \theta, \varphi)$. However, only one of the four two-point correlation functions is independent [51, 52]. Therefore, in the present work, only the two-point correlation function $P_{11}(r, \theta, \varphi)$ is considered. As $r$ approaches zero, $P_{11}(r, \theta, \varphi)$ approaches the value equal to the volume fraction of the particulate phase; as

*r* goes to infinity, it approaches the value equal to the square of the volume fraction of the particulate phase. At other values of the length of the line *r*, $P_{11}(r, \theta, \varphi)$ depends on the particle shapes/morphologies, the first-order global microstructural properties, the spatial arrangement of the particles, and morphological anisotropy. The microstructures studied in the present work are of extruded metal matrix composites. In such materials, the extrusion direction is the axis symmetry such that the microstructure observed in any metallographic plane containing the extrusion axis is statistically similar to that observed in any other plane containing the extrusion axis. This implies that the two-point correlation function measured in all planes containing extrusion axis would be statistically similar, and therefore independent of $\varphi$ [53].

*2.3.1.2 Lineal Path Distribution Function*

Consider a microstructure containing two phases, say particles (phase-1) and matrix (phase-2). The lineal path distribution function $L_{11}(r, \theta, \varphi)$ is the probability that a randomly located straight line of length r and angular orientation $(\theta, \varphi)$ is completely contained in the phase-1 [48, 54]. Figure 2.11 illustrates one such event that would contribute to this probability. Similarly, lineal path distribution function $L_{22}(r, \theta, \varphi)$ is the probability that a randomly located straight line of length r and angular orientation $(\theta, \varphi)$ is completely contained in the phase-2, i.e., the matrix. One can also define lineal path distribution function $L_{12}(r, \theta, \varphi)$ as the probability that a randomly located straight line of length r and angular orientation $(\theta, \varphi)$ intersects the interface between the particles and the matrix at least once [50]. Obviously, for any given values of r, $\theta$ and $\varphi$, the sum of these three lineal path probability functions must be equal to one.

$$L_{11}(r, \theta, \varphi) + L_{12}(r, \theta, \varphi) + L_{22}(r, \theta, \varphi) = 1$$

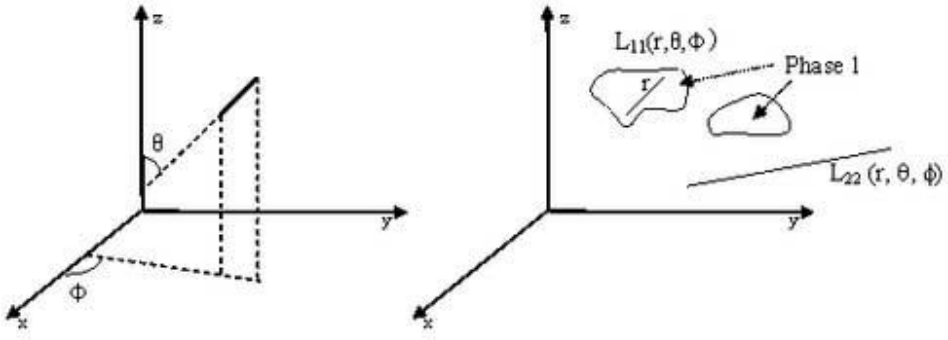



Figure 2.11: Schematic showing the relationship between r, θ and Φ for lineal path function $L_{11}(r, \theta, \Phi)$, where line of length r is completely contained in phase 1.

Therefore, only two of the three lineal path probability distributions are independent. Consequently, in this contribution, we will be concerned only with the estimation of $L_{11}(r, \theta, \varphi)$ and $L_{22}(r, \theta, \varphi)$. The function $L_{11}(r, \theta, \varphi)$ depends on the size, shape, and orientation distribution and the morphological anisotropy of the particles (phase-1). Note that $L_{11}(r, \theta, \varphi)$ does not provide any information about the spatial arrangement (clustering, etc) of the particles [54]. On the other hand, the function $L_{22}(r, \theta, \varphi)$ depends on the metric properties of the microstructure, the size, shape, and orientation distribution of the particles, their morphological anisotropy, *and* the *spatial arrangement and clustering* of the particles. Note that lineal path probability functions are independent of the two-point correlation functions, i.e., lineal path probability distributions cannot be calculated from the two-point correlation functions of the same microstructure. Thus,

lineal path probability distributions provide information about the microstructural geometry that is not contained in the classical n-point correlation functions.

### 2.3.2 Three-Dimensional Microstructure Reconstruction

Material microstructures are usually of three-dimensional nature and, therefore, characterization and visualization of *three-dimensional* (3D) microstructures is of primary interest. Even so, as most of the materials are opaque, the microstructural observations are generally carried out on the *two-dimensional* (2D) metallographic sections through 3D microstructural domains of interest. The microstructure observed in a metallographic section consists of intersections of the features in the 3D microstructure with the sectioning plane. Therefore, in a metallographic plane, the volumes (e.g., grains, voids, particles) in a 3D microstructure appear as areas, and the surfaces (e.g., grain boundaries, precipitate interfaces) appear as lines. Clearly, a 2D metallographic section does not contain all the information concerning the true 3D microstructural geometry. Although some microstructural attributes such as metric properties, two-point correlations, and lineal path probability distributions can be estimated from the measurement performed in the 2D sections using stereological techniques, the information concerning topological aspects of microstructure such as connectivity of particles, formation of particle chains and bands, percolation events, etc., cannot be obtained from independent 2D metallographic sections. Therefore, 3D microstructural reconstruction and visualization are of significant interest for understanding such aspects of 3D microstructural geometry.

Forsman was the first to reconstruct the 3D microstructure from serial sections [55]. He reconstructed the 3D microstructure of pearlite in steel from serial sections by projecting the image of each section onto cardboard layers. Forsman essentially developed the classical serial sectioning technique, which can be used to reconstruct a relatively small microstructural volume of an opaque material. This technique involves stacking of successively polished metallographic planes (sections) that are parallel to each other. The classical serial sectioning technique has been used in numerous investigations to study 3D microstructures of opaque materials [56-60], and it is quite useful for visualization of 3D particle/feature morphologies and short-range microstructural details at sufficiently high resolution. However, classical serial sectioning is not useful for quantitative characterization of topological attributes such as coordination numbers, and important descriptors of spatial arrangement of microstructural features such as higher order nearest neighbor distributions and radial distribution function due to serious bias (systematic error) resulting from edge effects [61-63]. Classical serial sectioning is also *not useful* for truly *unbiased* estimation of 3D grain/particle size distributions due to the same troublesome edge effects that create significant bias [64]. Further, the classical serial sectioning technique cannot be used to reconstruct a large volume of 3D microstructure (say ~ 1 $mm^3$) at sufficiently high-resolution (say ~1 μm). Therefore, it is not useful for characterization of long-range particle/grain clustering, formation of long particle chains and bands, etc. These limitations of the classical serial sectioning can be overcome by using the montage serial sectioning technique [65] developed at Georgia Tech in 1999. To generate a large volume of 3D microstructure at high resolution, one may first reconstruct a small microstructural

volume such as the one in Figure 2.12a, and then reconstruct many *contiguous* small volumes surrounding it, perfectly match their boundaries, and paste them together to generate a large microstructural volume, as shown in Figure 2.12b. The montage serial sectioning technique is available for such a reconstruction [65, 66], and in this research, it is applied to reconstruct a large volume of the 3D microstructures of the DRA composites. This technique involves the seamless joining contiguous single field of views (FOVs) to create a high resolution-large area montage (section). The specimen is successively polished to capture a number of such parallel montages (sections), which can then be combined to reconstruct the 3D volume of the specimen.

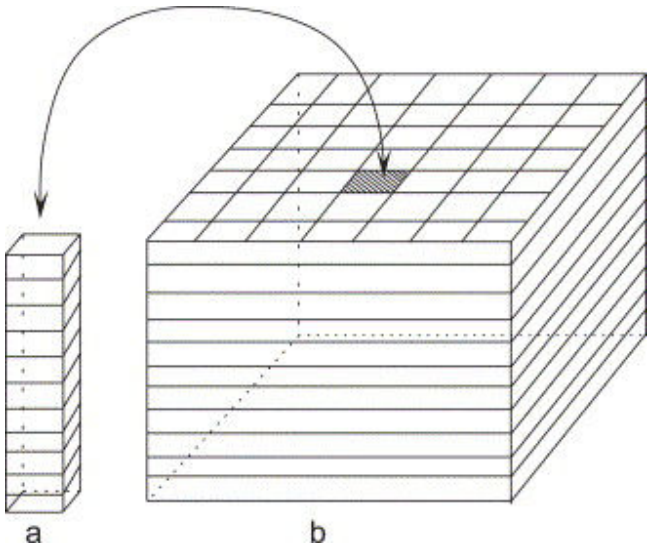



Figure 2.12: (a) Small microstructural volume element constructed from a stack consisting of one field of view in each serial section. (b) Large volume of microstructure obtained from contiguous small volumes such as those in (a) or by using montage serial sectioning.

The montage-based serial sectioning permits generation of significantly large volume ( few $mm^3$) of 3D microstructure at a *high resolution* ( 1 μm). For approximately

the same metallographic effort, montage-based serial sectioning yields a microstructural volume containing a *few thousand* particles/grains, which provides sufficiently large statistical sample for efficient, reliable, unbiased, and assumption-free direct estimation of 3D microstructural properties as well as for study of topological aspects of microstructure such as feature connectivity. Recently, montage serial sectioning has been implemented in a completely *automated* serial sectioning set-up that utilizes a robotic arm to move the specimen back and forth between the metallographic equipment (polishing, etching, etc.) and light optical microscope to generate the montage serial sections in a completely automated manner [67].

The present research concerns visualization of 3D microstructure from a stack of montage serial sections to observe short-, intermediate- and long-range spatial clustering and connectivity of SiC reinforcement particles in the specimens of discontinuously reinforced aluminum alloy (DRA) composites. Modern image processing and 3D image reconstruction techniques are used to reconstruct and visualize the 3D microstructure using volume and surface rendering techniques. These procedures are described in the following sub-sections.

## 2.4 Computer Simulations of Microstructures

As mentioned earlier, computational materials science provides a useful tool for prediction of materials properties via simulations of microstructures at relevant length scales and implementation of such microstructural windows in computational models and simulations of material behavior. Such simulations can be useful for performing parametric studies on the effects of microstructural geometry on the material properties if

and only if the simulated microstructures capture the relevant microstrucutral reality of the corresponding real materials. The central objective of this research is to develop a methodology for simulations of such realistic microstructures. Numerous computer simulation studies of random heterogeneous material microstructures have been reported in the literature [2, 3, 49, 50, 62, 68-82]. Important algorithms for computer simulations of heterogeneous microstructures include random sequential adsorption (RSA) algorithm [83], Metropolis algorithm [84], Boolean schemes [85], Gaussian random fields [86], simulated annealing process [87], Gibbs process [87] and Monte-Carlo techniques [88, 89]. The new methodology draws from some of these existing algorithms for simulations of random heterogeneous microstructures, and therefore, a brief review of these microstructure simulation techniques is given in this section.

### 2.4.1 Random Sequential Adsorption Algorithm

A popular technique used for microstructural simulations is random sequential adsorption (RSA) algorithm. In this algorithm, the features of interest, most commonly circles (2D) or spheres (3D), are added randomly and sequentially to the image plane or volume, respectively. Once the particle is placed in the image space it gets *adsorbed* and is not allowed to move throughout the simulation. If a new particle overlaps the existing particles in the image space, that particle is not placed at that location, and another random location is found till it can be placed on the simulation area/volume without overlapping the existing particles. The process is carried out till desired volume fraction and size distribution of the phase is achieved. For example, figure 2.13 shows a microstructure simulated in this manner. There is an upper limit on the volume fraction of

the second phase in the microstructure simulated in this manner, which is called jamming limit. For 3D microstructures containing monodispersed spheres simulated using the RSA algorithm, the jamming limit is 0.382 [82]. The main drawback of the classical RSA algorithm is that microstructures other than those having uniform random spatial arrangements cannot be generated via this method. Unfortunately, material microstructures often contain spatially clustered constituent phases (for example, see figure 2.4). Clearly, classical RSA algorithm cannot be used to simulate such microstructures. Nevertheless, the algorithm is useful in mimicking the randomness (stochastic nature) in microstructures. Consequently, in present research, the RSA algorithm (as well as some other algorithms) has been modified for simulations of heterogeneous microstructure where spatial arrangements are not uniform random.

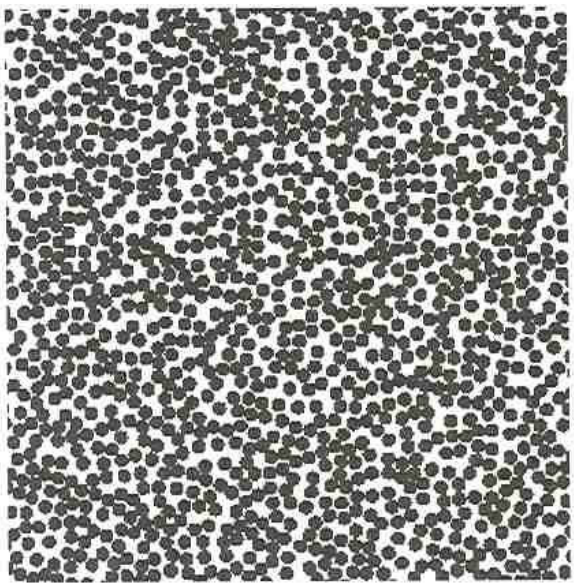


Figure 2.13: Image generated using RSA method [50]

2.4.2 Metropolis Algorithm

Another important algorithm for simulations of uniform random microstructures of impenetrable spheres/circles is Metropolis Algorithm (MA). This method is used to generate equilibrium ensemble of systems of interacting particles. In this scheme, the starting simulation space of the algorithm contains an arrangement of finite size particles and then every particle is individually displaced along each axis by an amount randomly picked from the interval [-δ, δ], where δ is the maximum step size. The displacement of the particle is accepted if it does not overlap with another particle otherwise it is discarded and new random number for the displacement is generated. Figure 2.14 shows three examples of starting arrangements namely, a square array, a hexagonal array and a random array.

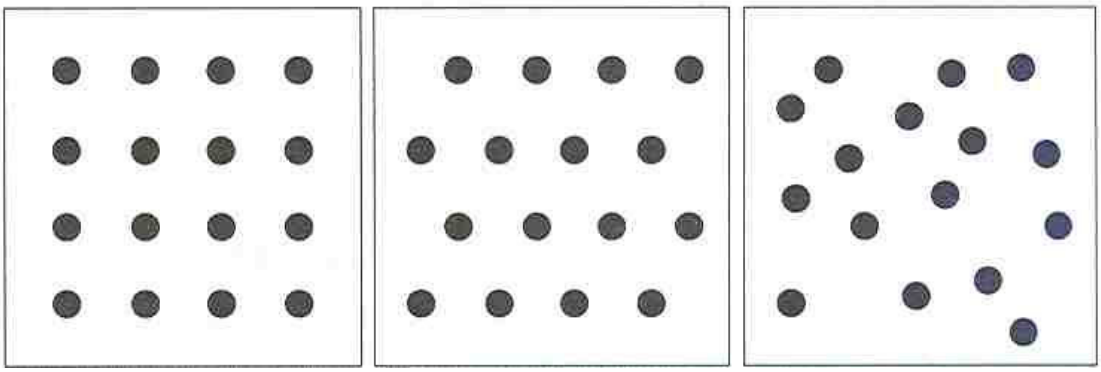


Figure 2.14: Starting configurations for Metropolis algorithm [50]

Since, via this method the simulation space can be sampled in a uniform way, Metropolis Algorithm is an excellent way of generating realizations that are in thermodynamic equilibrium. Also, higher volume fraction can be generated via this method as compared the RSA method. As MA involves step by step algorithm where

each particle is displaced individually and then the statistical correlation functions are extracted at each step, the method can be computationally extensive and time consuming. The computational power and time required to achieve the desired equilibrium will increase exponentially as the size of the realization is increased. For example, the simulation space required for generation of clustered microstructure having long-range heterogeneities shown in figure 2.4 is of the size such that the application of Metropolis Algorithm is not feasible.

### 2.4.3 Cherry-Pit Model

Random sequential adsorption and Metropolis algorithm are examples of *fully-impenetrable* or *hard-particle* models as they do not allow the overlaps of the constituents. These models have been used to study various systems such as liquids, glasses, thin films, membranes, powders and many other types of materials [49, 50]. However, if one wants to study the effects of topological connectivity of second phase particles on the properties of the materials, overlapping of the particles has to be permitted in the simulations. An example of a model where the particles are allowed to overlap is the *penetrable-concentric shell* model or the *cherry-pit* model [50]. In this model, each *d*-dimensional sphere of diameter D is composed of hard impenetrable core of diameter $\lambda D$, surrounded by a perfectly penetrable shell of thickness $(1-\lambda)D/2$, where $\lambda$ is an impenetrability parameter. The limits $\lambda = 0$ and 1 correspond, respectively to the case of fully penetrable sphere model, and the totally impenetrable sphere model. The effect of the degree of connectivity of the particle phase on the effective properties can be studied by varying the value of $\lambda$ between 0 and 1. Figure 2.15 shows a microstructure

constructed using cherry-pit model, where λD represents the impenetrable core and the outer shell of thickness (1-λ)D/2 is allowed to penetrate. Since this model has been developed for the second phase particles of circular (2D)/ spherical (3D) shapes, in present research, a modification of *cherry-pit* model is proposed to enable simulations of microstructures having complex realistic particle morphologies. In our modified cherry-pit model, instead of using the impenetrability factor λ, the area fraction of the particles which is overlapped by other particles is considered as the controlling parameter.

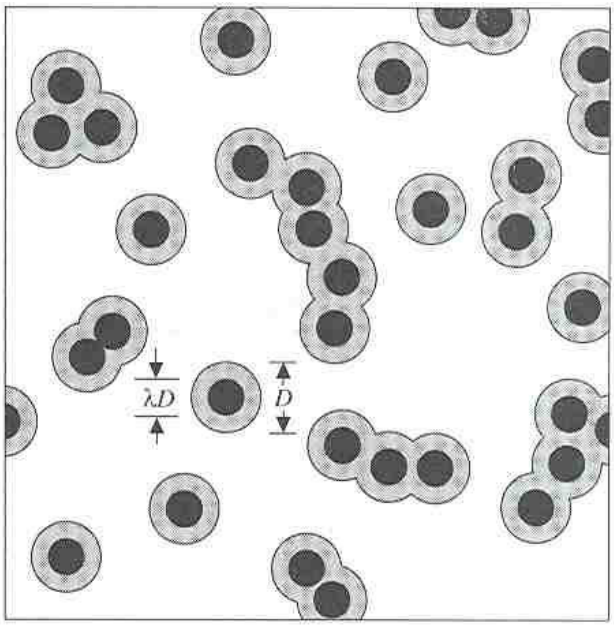



Figure 2.15: Microstructure constructed via cherry-pit model [50]

2.4.4 Voronoi Tessellations

A tessellation is a division of space using lines/planes based on a set of rules. Simulations of tessellations are useful for mimicking the grain structures in single-phase materials as well structures of cellular materials and foams. There is a wide variety of natural or man-made materials such as biological cells, plant organs, wood, polycrystals

and foams whose microstructures can be simulated using tessellations [49]. The abundance of cellular materials and the need to study the microstructure/properties of such materials has led to the development of tessellation models. Among the various algorithms for simulations of tessellated microstructures, the

Voronoi tessellation algorithm is perhaps the most popular one. Voronoi tessellations can be constructed by partitioning or tessellating the *d*-dimensional space into *d*-dimensional polyhedral cells. In literature, mathematicians refer to this technique as *Dirichlet tessellation* and the cells are known to condensed-matter physicists as *Wigner-Seitz* cells. In this model, one first creates a 2D or 3D simulation box and tessellate it by inserting a number of random (Poisson) points in it, each of which is the basis for a Voronoi polygon or polyhedron, which is that part of the space which his nearer to its Poisson point than to any other Poisson point. In practice, the Voronoi tessellation of space is done by first joining the Poisson points by straight lines and then bisecting these by plane surfaces. Figure 2.16 shows an example of a 2D Voronoi tessellation. The Voronoi representation of particle systems is a convenient way of identifying a particle's nearest neighbors and many other properties of this model, such as, two-point probability function can be determined either analytically or by computer simulations [49].

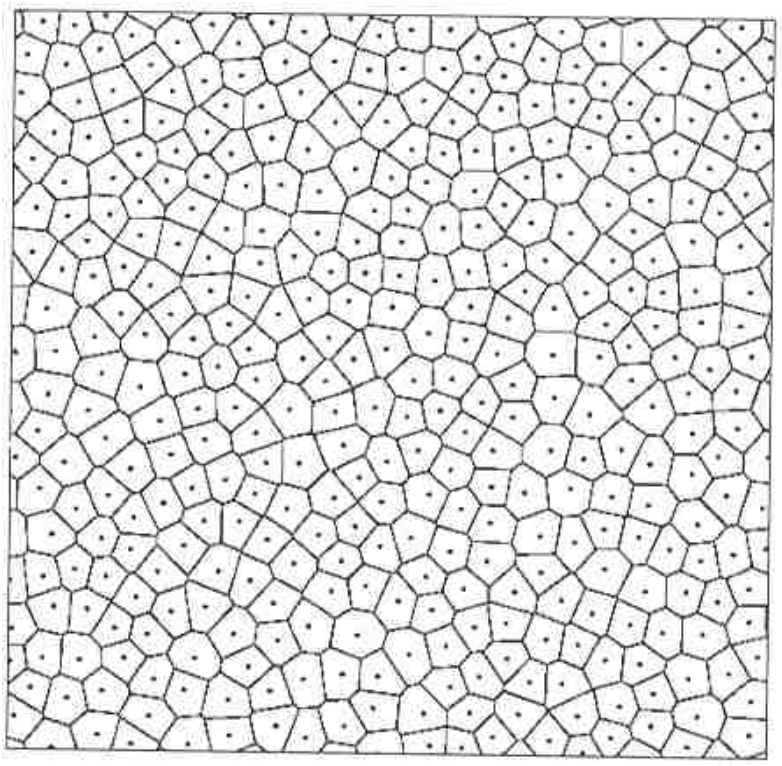


Figure 2.16: 2D Random Voronoi tessellations [50]

2.4.5 Gaussian Random Fields

A different type of simulation algorithm is required to simulate materials that are amorphous in nature. Such materials include amorphous alloys, microemulsions, carbonate rocks, and aerogels. To represent the microstructure of these types of materials, *random-field* models are employed. In these models, the interfaces between the phases are taken to be a level cut of a random field. The idea was originally developed by Cahn in 1965 and later formalized by Blumenfeld and Torquato (1993), according to which the generation of random-field is three step process: first, the source image is generated, followed by the coarse-graining of the source image and lastly, performing the level cuts on the coarse-grained image. Figure 2.17 shows 2D image generated using this model.

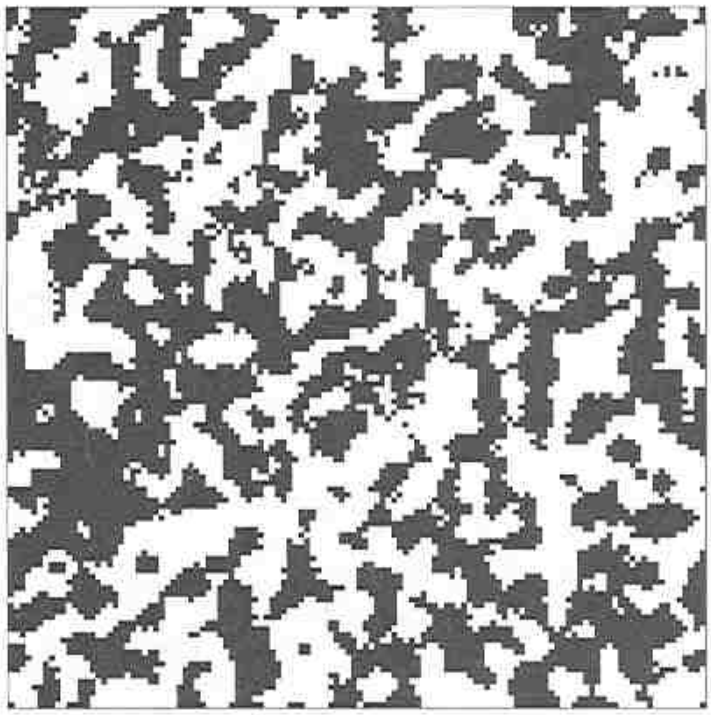


Figure 2.17: 2D microstructure generated using Gaussian random field [50]

2.4.6 Monte Carlo Methods

The aforementioned simulation models can be used to generate various types of microstructures depicting morphologies ranging from fully impenetrable spheres to cellular structures to amorphous alloys. Nonetheless, these techniques are not useful for simulations of microstructures with any specified statistical representations such as n-point correlation functions. Monte Carlo methods are useful for simulations of such targeted microstructures [50]. Monte Carlo techniques involve the stochastic generation of simulation images (both spatial position and orientation of second phase particles) and sampling them according to the given probability function(s). The process is carried out till the desired values of correlation functions are achieved. The simulated realizations

can be compared with the real microstructures using a variety of statistical correlation functions, such as, n-point probability functions, lineal path distributions, radial distributions, nearest neighbor distributions, etc. Among these, two-point probability and lineal path correlation functions are employed in this research.

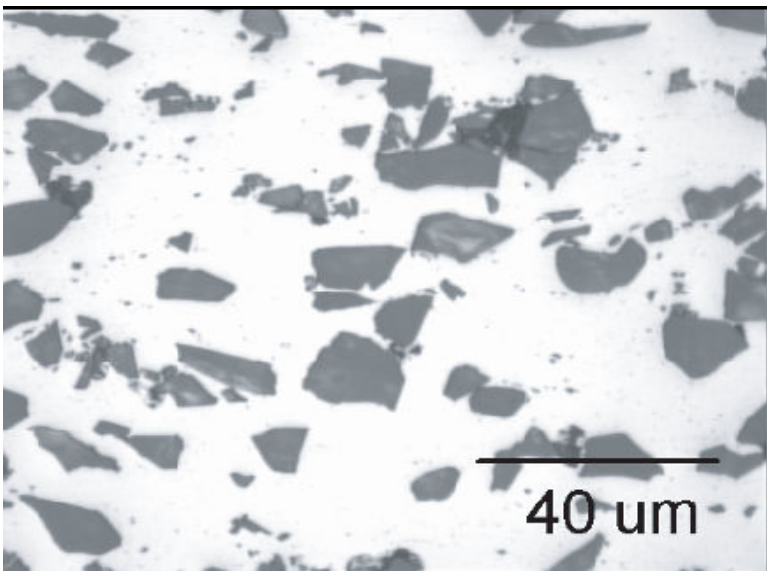



Figure 2.18: Micrograph depicting the complex shapes of SiC reinforcement particles in the DRA composites.

As can be seen from this review, the existing models for the simulation of microstructure have several shortcomings, including the assumption of idealized simple particle/features shapes such as spheres and ellipsoids in 3D and circles and ellipses in 2D or incorporation of unrealistic arbitrary digitized particle shapes. As evident from Figure 2.18, the complexity of the real reinforced particles can not be approximated by assuming shapes such as circles or ellipses. Further, in many cases, the simulated microstructural segments are too small to account for important long-range spatial patterns and heterogeneities in the microstructure, or to serve as representative volume elements (RVEs) of the microstructure in computational models for material behavior.

For reliable predictions of material properties, it is imperative that the simulated microstructures represent the corresponding relevant microstructural reality. Apart from the first-order attributes such as relative amounts of different phases (volume fractions) and their number densities, etc., feature/particle shapes and morphologies, spatial arrangements and clustering of features and morphological anisotropy are also important aspects of microstructural geometry that affect the material properties, and therefore should be incorporated in the simulated microstructures. Accordingly, there is a need to develop a methodology to simulate such realistic microstructures. The simulation methodology developed in this research enables simulations of realistic microstructures that incorporate realistic complex particle/feature shapes; admits controlled non-uniformities/clustering in spatial distributions of features; permits partial anisotropic morphological orientations of microstructural features; closely matches experimentally measured attributes (spatial correlation functions, orientation distributions, size and shape distributions, volume fraction, etc.) of the corresponding real microstructures; and can efficiently generate sufficiently large segments of microstructure that contain short-range (of the order of particle/feature size), intermediate-range (5–10 times particle/feature size), and long-range (100 times the particle/feature size) microstructural heterogeneities and spatial patterns. The new methodology is described in detail in chapter 4.

# CHAPTER 3

# EXPERIMENTAL WORK

The central objective of the present research is to develop the methodology for computer simulations of realistic microstructures via its applications to the microstructures of SiC particles in Al-alloy matrix composites and TiB whiskers in boron modified Ti-alloys. The simulations of realistic microstructures require detailed experimental quantitative microstructural data on a few specimens processed under differ processing conditions. In the present case, Dr. J. Spowart and Dr. S. Tamirisakandala carried out the experimental work on the materials processing at the Air Force Research Laboratory (AFRL), Wright Patterson Air Force Base, whereas the metallography, three-dimensional microstructure reconstructions, and digital image analysis and stereology based detailed quantitative microstructure characterization were performed at Georgia Tech. The microstructure characterization work also involved development of a digital image analysis based technique for unbiased estimation of three-dimensional lineal path probability distributions from vertical metallographic sections [90]. The experimental work associated with the present research is described in this Chapter. The next section describes the powder metallurgy based processing of the materials under study, namely, Al-SiCp composites and boron modified titanium alloys and composites. The metallography and three-dimensional microstructure reconstruction are discussed in the subsequent sections.

## 3.1 Materials and Processing Details

As described in the previous chapter, there are several ways of processing both discontinuously reinforced aluminum composites and boron modified titanium alloys and composites. In the present research, Powder Metallurgy (P/M) routes have been used to fabricate the two materials of interest. The materials processing details are given in the following subsections.

### 3.1.1 Al-$SiC_P$ Composites

The DRA samples used for this research were fabricated by Dr. J. Spowart at the Air Force Research Laboratory, Materials and Manufacturing Directorate, Wright-Patterson AFB, Ohio via the P/M route. Al-6061 matrix alloy powders (Al-0.27Cu-0.26Fe-0.97Mg-0.56Si) were initially screened to –325 mesh (<45 μm) prior to blending with SiC reinforcement. In order to reduce agglomeration due to electrostatic forces, the matrix and reinforcement powders were suspended in slurry using 1-butanol as the solvent during the blending stage. After blending, the powders were dried, re-screened to –325 mesh and placed in an extrusion can. In order to remove all traces of residual solvents from the powders an elevated temperature vacuum de-gas treatment was used. After removing the solvents the cans were sealed for compaction and extrusion. Extrusion was carried out at 450 $^0$C, with an extrusion ratio of 25:1 (round: round), followed by an air cool [91, 92].

Three different samples of the DRA composites, each with different matrix to reinforcement particle size ratio (PSR), are used for this work. As reported in the previous chapter, PSR is an important processing parameter that controls the

homogeneity of the reinforcement particle distribution in composites manufactured via P/M route [1]. Increasing the PSR leads to a reduction in the combined surface area of the matrix alloy particles and as this area becomes insufficient for a uniform arrangement of reinforcement particles, clusters of second phase particles are formed in-between the larger matrix particles. All the Al-$SiC_P$ composites studied during this research contained SiC particles of 13.4 μm median diameter ($d_{50}$), but 6061 Al-alloy matrix powders of three different median diameters, 26.8, 41.54 and 108.54 μm were used to produce composites having particle size ratios (PSR) of 2, 3.1 and 8.1. This design of experiments yields a set of microstructures having different degrees of spatial clustering of SiC particles but the same volume fraction, mean size, and size/shape distribution of the SiC particles as well as the matrix alloy chemistry. These specimens have been utilized for development and applications of the stereology and image analysis based techniques for quantitative characterization of the spatial clustering. The resulting data have been used to correlate the spatial clustering with the processing parameter of PSR via simulations of *realistic* microstructures, which is discussed in the later chapters.

### 3.1.2 Boron Modified Titanium Alloys

The SiC particles in the DRA composites have isotropic morphological orientations. On the other hand, in numerous microstructures the constituent phases have preferred morphological orientations leading to partially or completely anisotropic microstructures. In the present work, the microstructures of TiB whiskers in the boron modified Ti-alloys have been utilized to further develop the simulation technique to enable the computer simulations of partially or completely anisotropic microstructures.

The experimental measurements have been performed on boron modified Ti-6Al-4V alloy produced via a pre-alloyed powder metallurgy approach at Crucible Research Corporation, Pittsburgh, PA. The Ti-6Al-4V-1.0B powder of -100-mesh size (average particle size of 150 μm), was packed inside a thick-walled can of Ti-6Al-4V, vacuum outgassed at 300°C for 24 hours, and sealed. The can was heated to 1200°C, soaked for 1 hour, and then blind-die compacted in an extrusion chamber heated to 260°C. The billet height was reduced by about 30% at a ram speed of 6.35 mm s-1, and the compact was held at a pressure of 1400 MPa for 180 seconds and subsequently air-cooled to room temperature. A second billet blind die compacted by the same method was subsequently hot extruded at 1100°C with an extrusion ratio of 16.5:1, at a ram speed of 6.35 mm s-1, and air-cooled to room temperature. Sections of specimens were then taken from the compacted billet and extruded rod. The extrusion and heat treatment experiments were carried out by Dr. S. Tamirisakandala at the Air Force Research Laboratory, Wright-Patterson Air Force Base, Ohio.

## 3.2 Metallography

### 3.2.1 Metallography of DRA Composites

The 2D metallographic samples were prepared by sectioning the longitudinal cross-section containing the extrusion axis. These specimens were mounted in bakelite and then tightly fixed in a holder in order to preserve their orientations. The grinding steps involved polishing on polishing-papers (320 to 600) followed by fine polishing using diamond based oil compounds (15 μm to 1 μm). During these steps, glycerol and

oil base lubricants were used to avoid the formation of surface film. After the 1 μm polishing, the final step involved the use of 0.05μm colloidal silica. The specimens were immediately rinsed and methanol and dried after each of these steps. Figure 3.1 shows a micrograph for specimen with PSR 8.1 obtained by the above process. The detailed analysis of variation of microstructures with respect to change in PSR is discussed in next chapter.

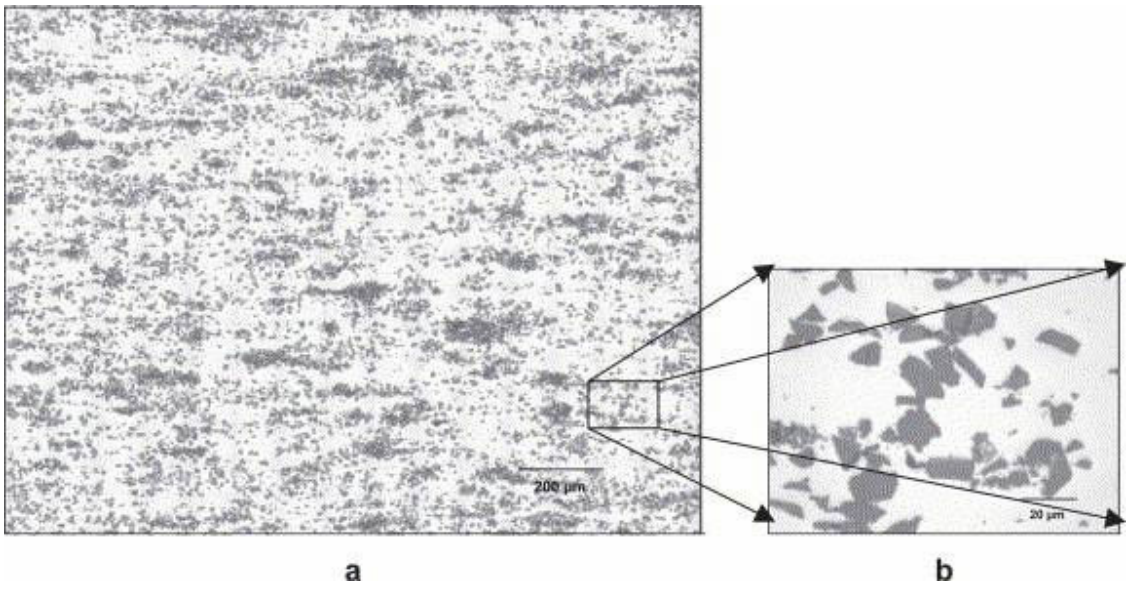
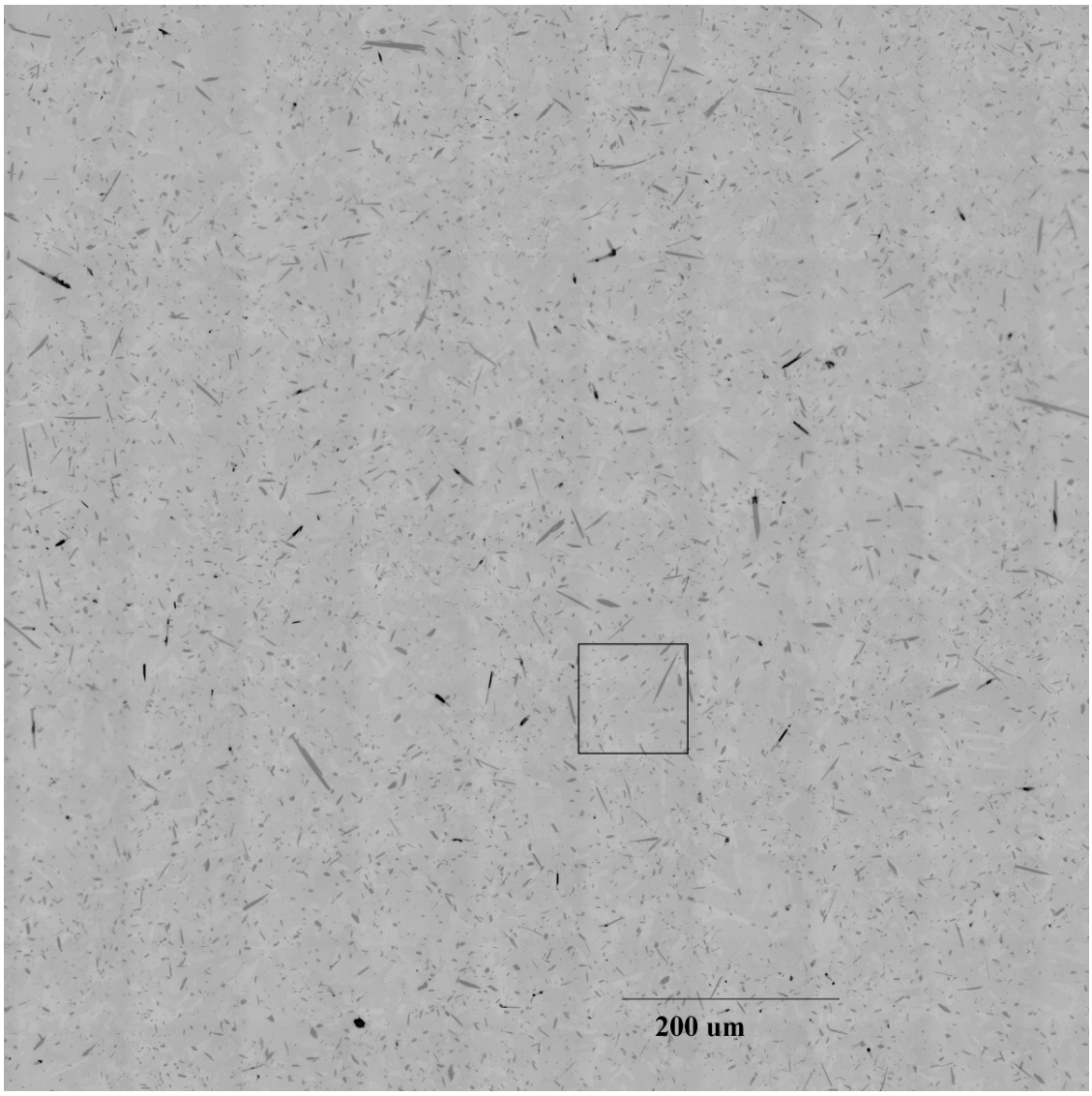



Figure 3.1: (a) Micrograph showing microstructure of specimen with PSR 8.1, (b) high resolution image depicting the complex shapes of SiC particles.

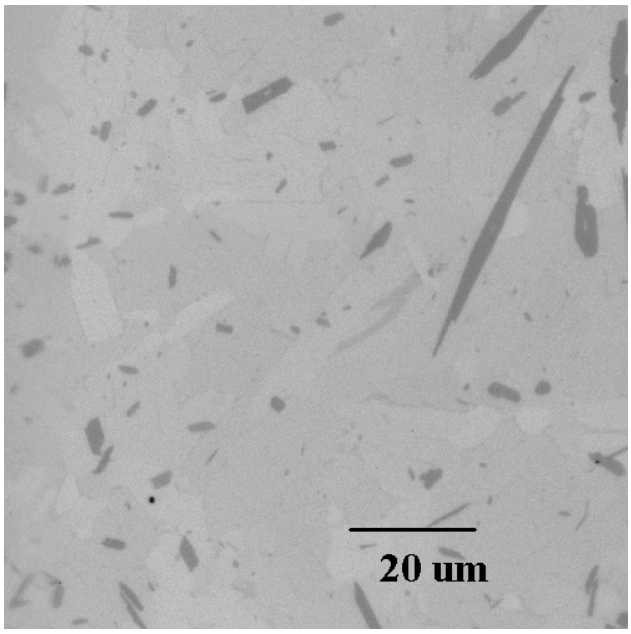
3.2.2 Metallography of Boron Modified Ti-Alloys

The metallography of boron modified Ti alloys was performed by Dr. Scott I. Lieberman as a part of his PhD thesis work [32]. After mounting the specimens of compacted billet and extruded rod in thermoplastic, grinding and polishing procedures were carried out. These procedures took into account the hardness difference between the relatively soft Ti-6Al-4V alloy matrix and the relatively hard TiB whisker reinforcement.

Final surfaces finish of 0.05μm is used to examine the microstructure. Figure 3.2 shows one such micrograph for compacted boron modified Ti-6Al-4V alloy. The differences in microstructures between compacted and extruded specimens are discussed in detail in the next chapter.




(a)




(b)

Figure 3.2: (a) Micrograph showing microstructure for compacted boron modified Ti-6Al-4V alloy. (b) High resolution image showing the TiB whiskers.

### 3.3 Statistical Descriptors

Statistical descriptors are useful for characterization of spatial arrangement and heterogeneity of microstructural features. As mentioned earlier, two-point probability functions and lineal path distributions are used in this work to characterize the microstructures of the composites using 2D montages. The following sub-sections details the algorithms used to compute the statistical correlation functions using the 2D binary images of the microstructures of the materials under study.

### 3.3.1 Two-Point Probability Functions

Recently, a robust digital image analysis and stereology based technique has been developed at Georgia Tech for estimation of direction dependent two point correlation functions of any three-dimensional microstructure from the measurements performed on vertical metallographic sections [53]. The technique permits precise and automatic estimation of the two-point correlation functions at distances ranging from 1 μm to 1000 μm at a resolution on the order of 0.5 μm; the correlation functions can be estimated at all discrete line orientations in the vertical planes. The technique involves the measurement of number of times a line segment of length *r* and angle θ has both its end in the phase 1 divided by the total number of times that line segment has been placed in the image frame to calculate the value of $P_{11}[r, \theta]$. The computer code for calculating two point probability functions is provided in Appendix C. In the present work, the technique has been applied for estimation of the two-point correlation functions of the DRA composites and boron modified Ti-alloys. These functions have been utilized in capturing the spatial heterogeneities and the morphological anisotropies in the materials of interest. Figure 3.3 shows an example of two point probability function measured for the DRA composite with PSR 8.1 in the extrusion direction ($\theta = 0$). The detailed results of the application of the two-point probability functions to all the materials under study are provided in the next Chapter.

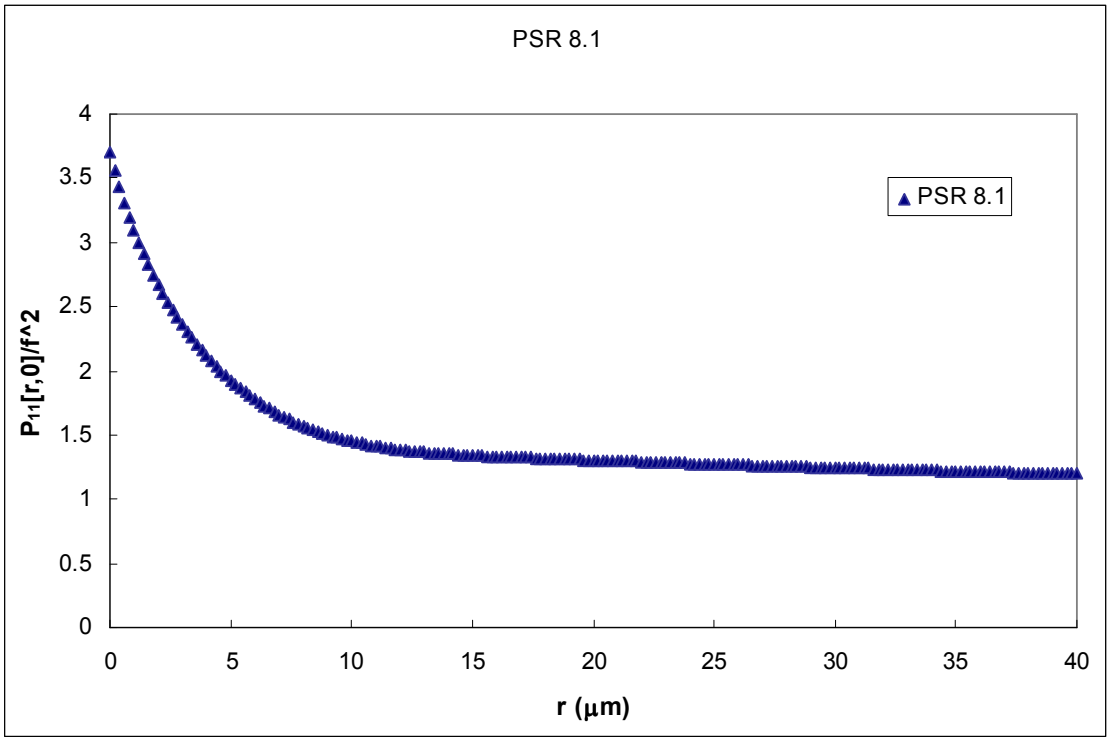



Figure 3.3: Two-point probability function for the microstructure of the DRA composite with PSR 8.1.

### 3.3.2 Lineal Path Probability Distributions

Computations of lineal path probability distributions are required for representation of short range, *intermediate range, and long-range* spatial patterns, correlations, anisotropies, and heterogeneities in microstructures. Nonetheless, all the earlier experimental data on lineal path probability distributions observed in metallographic planes involved measurements on just one or a small number of "single field of view" micrographs, and concerned measurements along just one direction. Consequently, the earlier data do not contain information concerning intermediate and long-range spatial patterns, or preferred orientations. Further, the short-range data have large random sampling errors due to measurements on just one or few micrographs (small

sample size). In addition, all earlier data also have a bias (systematic error) of unknown magnitude *due to the edge effects*. All of these limitations can be removed by using large-area-high-resolution digital image montages for the measurements of the lineal path probability distributions.

As a part of this research, a visual C++ program has been developed to compute the lineal path probability distribution functions. The binary images of the montages are used as an input for the code. The basic algorithm for the computer code is as follows. The program reads the binary image and asks the user for the input to the maximum length (lm) to which the lineal path function is to be measured. The input binary image should be at least of this length lm (in both length and breadth). In the present work the lineal path functions were measured for r valued up to 500 um. The program allocates a 2D array to two variables (L[1][r] and L[2][r]) for storing the number of occurrence of each $L_{ii}(r, \theta)$ events at a given distance and angle. The program then moves to the next step of measurement of lineal path function. Procedure for the measurement of lineal path in x-direction is described as follows. First step is to measure the number of occurrences of $L_1(r, 0)$ and $L_2(r, 0)$ and store them in their respective arrays. The code starts by scanning the image line by line (horizontal lines for x direction function). Each pixel in the given line is read for its grey scale value starting from the first pixel. The code keeps on moving to the next pixel until the grey scale value changes. This gives us the number of pixels contained completely in that particular phase (denoted by the grey scale value). Now the values of the arrays L[1][r] or L[2][r] are augmented depending on the phase in which the line segment was observed. For example, if 10 pixels of phase 1 are encountered, the value of L[1][1] is increase by 10, L[1][2] by 9 (since we can place 9

line segments of length 2 pixels on a line of 10 pixels) and so on. The code then continues along the same line till another phase change or end of line is reached. This gives us the L[i][r] values for a single line and the procedure is repeated for all the horizontal lines in the image to get overall L[i][r] values in the x direction. In order to calculate the total number of possible line segments (for each r) in the image frame, the values are calculated for a single line and then multiplied by the number of lines (i.e. y pixel dimension of the image in case of lineal path in x direction). For any given r the total number of line segments which can be put on line of length x would be x-r. These values depend only on the dimensions of the image and are independent of its content. Finally, the lineal path probability function for the given direction is calculated by dividing the occurrence of a specific $L_{ii}$ for a length by the total number of length segments possible in the image frame for that length. These probabilities are the desired lineal path functions in the x direction. Similar procedure is used to compute the lineal paths along other directions. Analytical equation for calculating approximate values of lineal path function for a system of fully impenetrable mono-dispersed circles of radius R as formulated by Lu and Torquato is given by

$$L(r) = (1-\eta)\exp\left[-\frac{2\eta}{\pi(1-\eta)}\frac{r}{R}\right].$$

In this equation L(r) is the lineal path function for the matrix and in the case of totally impenetrable circles η is equal to the volume fraction of circles [93]. In order to validate our algorithm a random homogenous image of circles with area fraction 0.2 was generated. Figure 3.4 compares the lineal path function as obtained by the analytical

method with the one measured directly from the image using our computer program. As can be seen from the plot they show a good match.

The present algorithm has proved to be effective in number of ways. First, it can calculate the lineal path function for particles and matrix at the same time. There are other algorithms in literature which can only find the lineal path in particles, which may not be enough in a lot of cases, for example clustering of particles inside the matrix will not be depicted by the lineal path function of the particles alone. For that we need the lineal path function inside the matrix, which is delivered by this computer code along with that of the particles. Secondly, this algorithm can measure lineal path functions up to large distances without consuming extra time. Since the code is scanning each line completely the distance up to which lineal path is measured is only limited by the dimension of the image, hence for the same computational effort large scale measurements can be easily performed. Finally, this algorithm provides us lineal path function in the $45^0$ direction along with the x and y directions, and it can be easily extended for computations along any other direction. The details of application of this technique to lineal path distributions for materials under study are provided in the next Chapter.

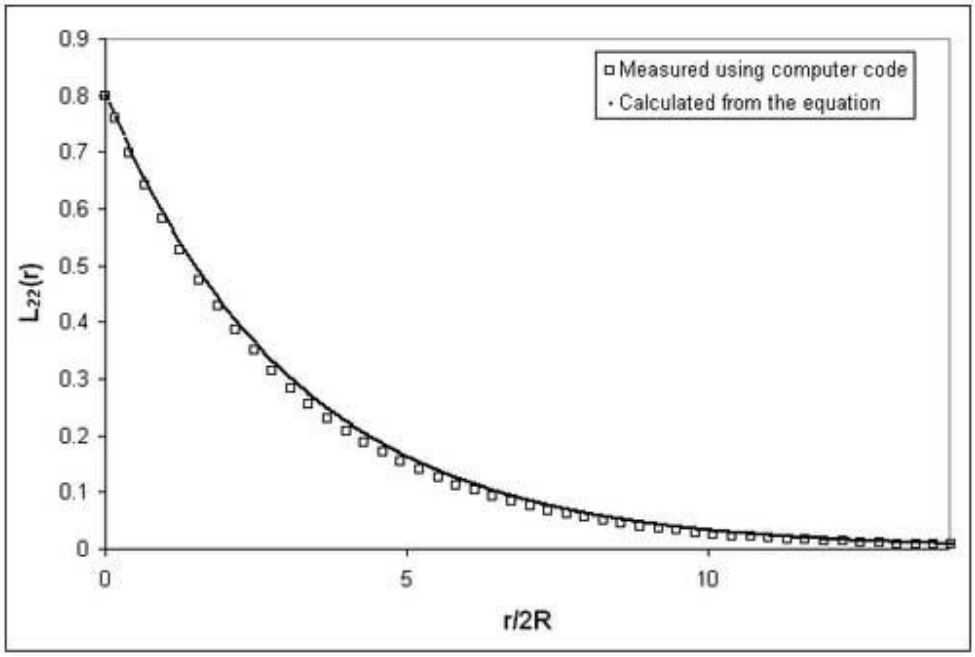



Figure 3.4: Comparison of matrix lineal path functions as obtained from analytical equation and computer program.

### 3.4 Reconstruction of 3D Microstructures

To generate a large volume of 3D microstructure at high resolution, the montage serial sectioning technique has been applied to reconstruct a large volume of the 3D microstructures of the DRA composites. First a "montage" of 125 contiguous microstructural fields observed at a high magnification (500 X for the present microstructure) is created by using the large-area high-resolution montage procedure developed by Gokhale et al. [62, 69, 94]. To create a montage, a field of view (FOV) is arbitrarily chosen in the region of interest in a metallographic plane, and the image of this field of view is stored in the memory of the image analysis computer as an image file. The right border (of 60 pixel width) of this image is recalled on the left edge of a blank image. This semi blank image is then displayed along with the live image. This results in

a superimposed image on the left border of the screen (of the previous right border and live image) with rest of the screen having the live image. The automated microscope stage is then programmed to move so that the right border of the live image moves to the left border and gives a reasonable match with the superimposed image. The physical movement of the automatic stage has a large least count and thus cannot achieve perfect match with the previous image. Therefore, an image cross-correlation function based technique has been used for automatic pixel-by-pixel matching of the overlapping borders of the two images, which results in a match of the first and the second image with an accuracy of one pixel (in the present case, the pixel resolution was 0.2 μm, and microscope resolution was 1 μm). The second image is then stored in the computer memory as another image file. All successive contiguous images are grabbed by using the same procedure and finally a seamless montage of large number contiguous microstructural fields is created. Figure 3.1a shows such a montage of 125 fields of view (FOV), which has been compressed for display. Each region of this montage has a high resolution of the image shown in Figure 3.1b. Therefore, the montage is a microstructural image of a large area (~ 1.92 $mm^2$) having a high resolution. In the present work, image analysis was performed with a KS-400 image analysis system from Kontron. However, several other commercial image analysis systems also have the required capabilities. The computer codes for creating the montage were written in a language similar to C++ in a platform provided by the image analysis software (KS-400).

Once the montage of the first serial section is created and stored in the computer memory, small thickness of the specimen is removed by polishing, and then a second montage is created at the region exactly below that in the first metallographic plane. In

the present study, this polish–montage–polish procedure was repeated to obtain stack of 100 montage serial sections with an average distance of 1 μm between two successive sections. Micro-hardness indents were used to locate the exact region of interest in successive serial sections and to measure the distance between consecutive serial sections [61, 95]. Figure 3.5a shows the geometry of Vickers-hardness indenter used in this research and Figure 3.5b shows an impression of this indenter on the DRA composite microstructure.

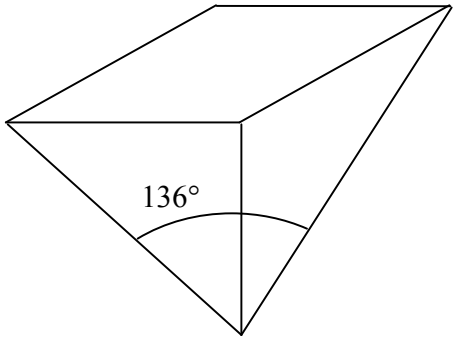



Figure 3.5: (a) Schematic showing the geometry of Vickers-hardness indenter.

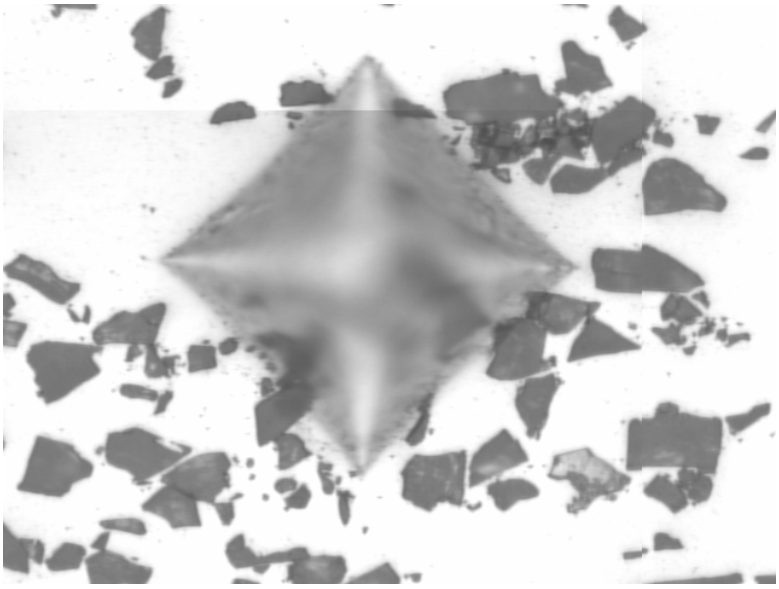


Figure 3.5: (b) Impression of the Vickers-hardness indenter on the microstructure.

An important practical problem in the reconstruction of a 3D microstructure from serial sections is that the successive serial sections may not be precisely aligned; they may have some translational and rotational displacement with respect to each other. In the present study, in spite of adjusting the microscope stage, the montages of the consecutive serial sections were often displaced by about ±10 pixels and ±5°, and therefore, it was essential to precisely align successive serial sections. Alignment can be achieved by locating two common points (in the present case, micro-hardness indents were used for this purpose) in the two consecutive serial sections and translating one image until the first common point is aligned in the two images. Then the image is rotated about this point until the second common point is also aligned. In the present case, this was accomplished by using 3D image analysis software Voxblast 3.10 in which the images of the montage were digitally translated and rotated until they were exactly aligned to the respective previous sections.

### 3.4.1 Reconstruction and Visualization of 3D Microstructures

The stack of aligned serial sections essentially constitute a volume image data set similar to those encountered in X-ray computed tomography and magnetic resonance imaging (MRI). The steps involved in the 3D visualization of such data sets are as follows.

• data generation (in the present case, serial sections);

• pre-processing such as image alignment, grid regularization, image enhancement, and interpolation;

• rendering of 3D images.

The 3D microstructural visualization can be achieved either by surface rendering or by volume rendering. Surface rendering involves rendering of the iso-surface of the region of interest (ROI) from the volume data, whereas volume rendering is the rendering of *all* volume data by specifying opacity and color of each voxel (3D pixel). Surface rendering leads to reduction in the size of the data set because only the surface data are retained. The surface rendering requires fitting of a surface in the volume data. Numerous algorithms are available for surface rendering, including the contour connecting algorithm [96] and the marching cube algorithm [97]. In the present work, the marching cube algorithm has been used for surface rendering of 3D microstructures of the DRA composites. In the process of volume rendering, all voxels are visualized by specifying a mapping between the rendered image intensity and voxel intensity. In the present work, a ray-casting algorithm [98] has been used for volume rendering of the microstructural images. All 3D image rendering work was done by using image analysis software Voxblast 3.10.

In the present work, 3D microstructural visualization of the DRA composites has been done using 100 montage serial sections; each montage serial section containing 100 contiguous microstructural fields grabbed at 500×. Each reconstructed 3D microstructural segment has a volume of 0.192 $mm^3$ at a resolution of 1μm. Thus, a relatively large 3D microstrucutral volumes have been reconstructed at sufficiently resolution. Therefore, the resulting 3D data sets are useful for characterization and visualization of both short range and long-range attributes of the SiC reinforcement phase. Figure 3.6a shows a stack of 20 aligned montage serial sections for the 8.1 PSR composite microstructure, where each serial section is a digitally compressed montage of 100 contiguous microstructural fields. Figure 3.6b shows the magnified view of the small bordered region in Figure 3.6a and similarly Figure 3.6c shows the high resolution image of the bordered region in the Figure 3.6b. Figure 3.7 depicts a small segment of *surface*-rendered reconstructed 3D microstructure of the 8.1 PSR specimen. In the present work, the experimental 3D microstructural reconstructions have been utilized in studying the information concerning topological aspects of microstructure such as connectivity of particles, formation of particle chains and bands (clusters) and also the anisotropies of reinforced particles in the 3D space. These aspects of the 3D microstructures of the composites of interest are discussed in detail in the next chapter.

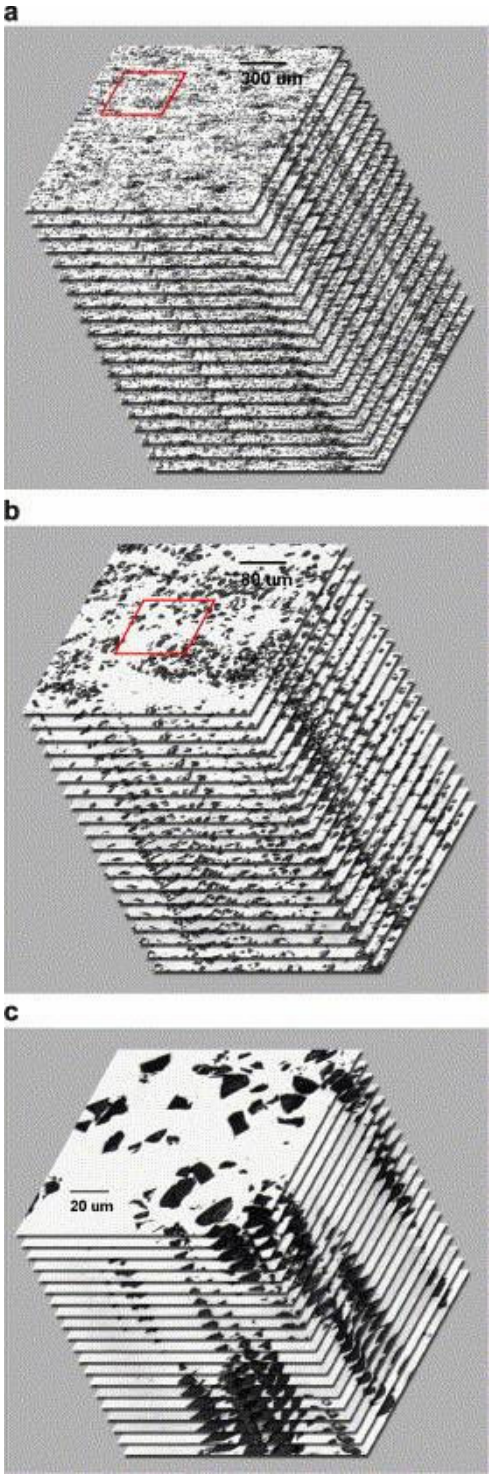



Figure 3.6: (a) Stack of 20 montage serial sections for the 8.1 PSR specimen microstructure. (b) The magnified view of the small bordered region of the stack of 20 montage serial sections for the 8.1 PSR specimen microstructure in (a). (c) The magnified view of the small bordered region of the stack of 20 montage serial sections for the 8.1 PSR specimen microstructure in (b).

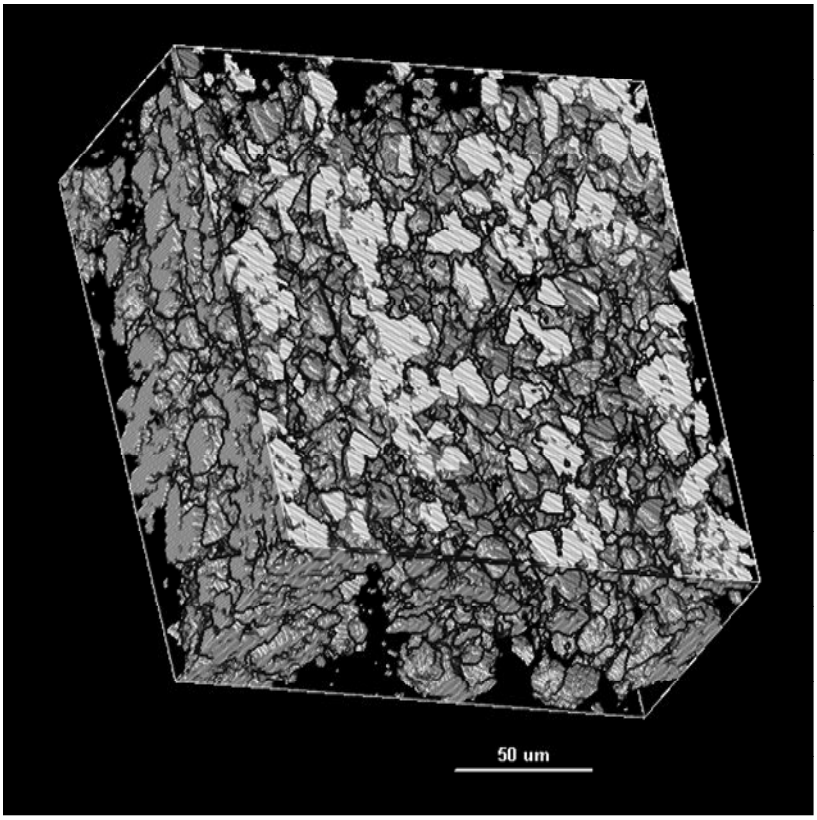



Figure 3.7: Small segment of surface rendered three-dimensional microstructure of DRA specimen with PSR 8.1.

# CHAPTER 4

# EXPERIMENTAL RESULTS

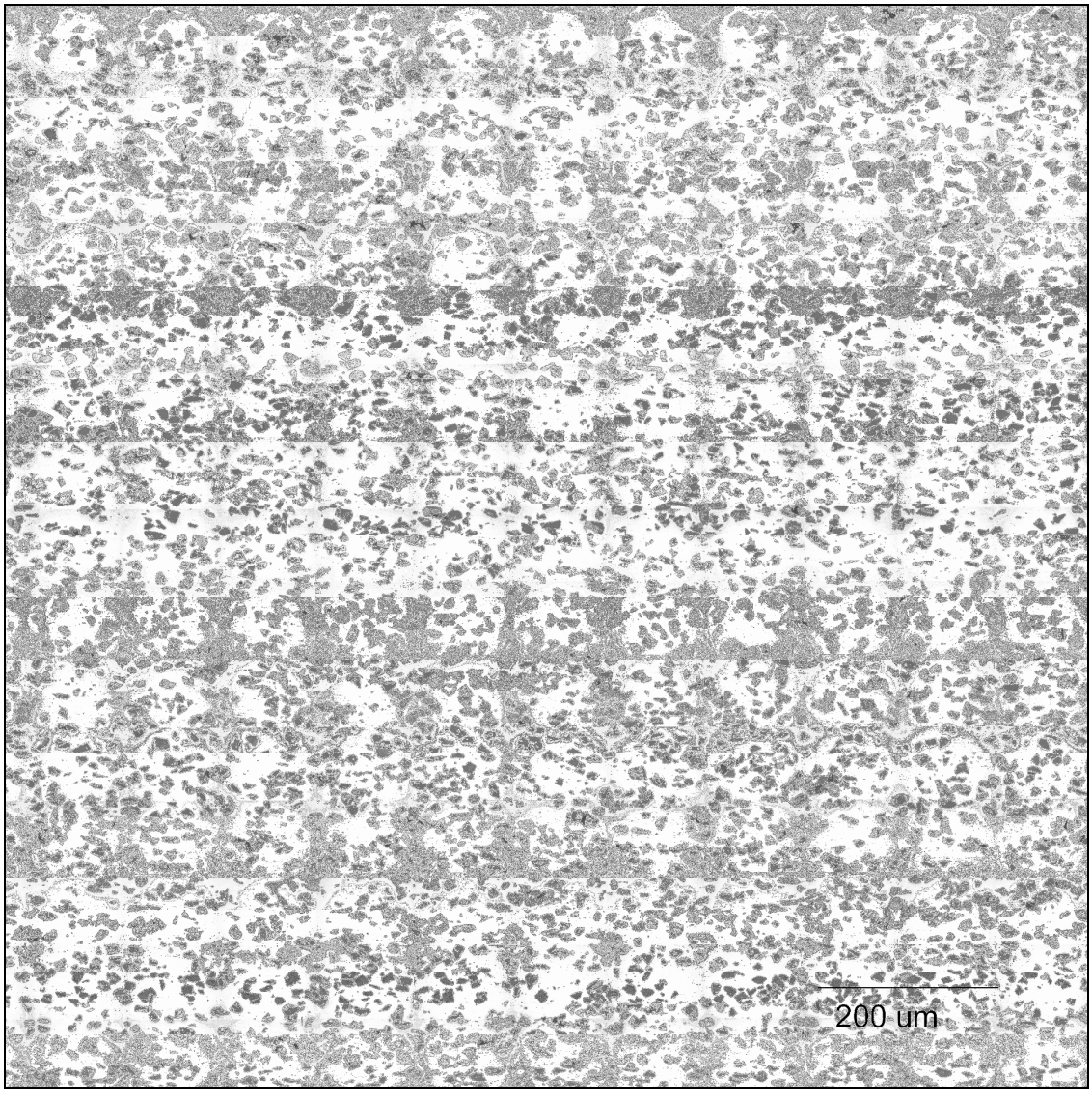
An important objective of this work is to develop a methodology to create 'realistic' simulations of microstructures. This technique has been developed via its applications to the microstructures of SiC particles in Al-alloy matrix composites and TiB whiskers in boron modified Ti-alloys. Creation of 'realistic' simulations requires detailed quantitative microstructural data on a few specimens. The experimental work necessary for obtaining the required data has been described in detail in the previous Chapter. The results of these experiments and the data are presented in this Chapter. These data are utilized in the next Chapter for simulations of realistic microstructures of SiC particles in the DRA composites and TiB whiskers in the boron modified Ti-alloys.

## 4.1 Qualitative Microstructural Observations

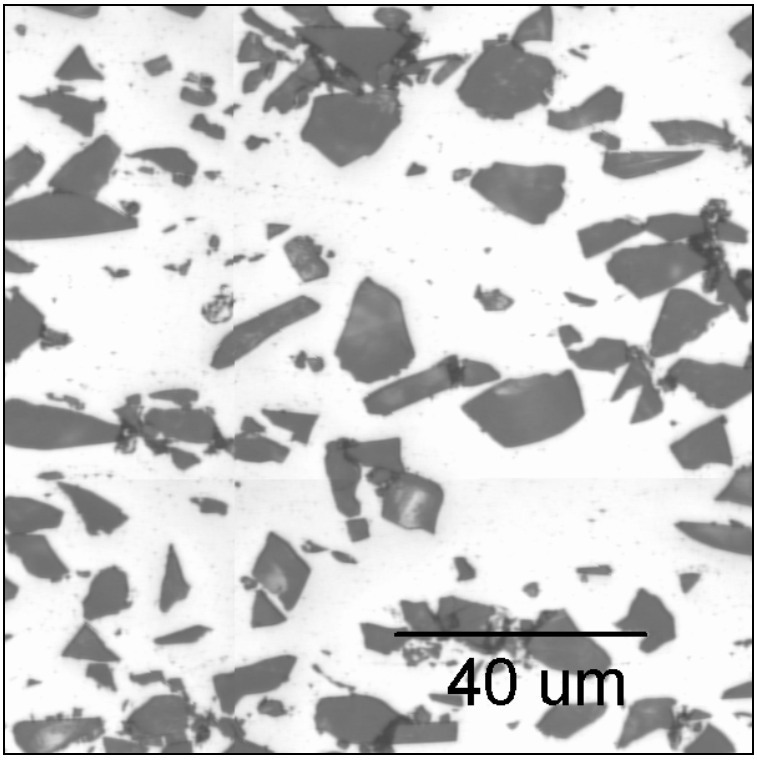
As described in the previous chapter, montage technique was used to construct high resolution, large area micrographs for the DRA composites and boron modified titanium alloys. These montages enable us to study the short-range, mid-range and long-range microstructural geometry and spatial patterns. The montage creation for the boron modified titanium alloys was done by Dr. Scott I. Lieberman as a part of his PhD thesis work. The following subsections qualitatively describe the 2D microstructures of the materials of interest.

### 4.1.1 Al-$SiC_P$ Composites

Three different samples of the DRA composites, each with different matrix to reinforcement particle size ratio (PSR), are used for this work. As reported in the previous chapter, PSR is an important processing parameter that controls the homogeneity of the reinforcement particle distribution in the composites manufactured via powder metallurgy route [1]. The Al-$SiC_P$ composites studied in this research have particle size ratios (PSR) of 2, 3.1 and 8.1. Figures 4.1 to 4.3 show micrographs for three DRA samples with PSR 2.0, 3.1 and 8.1, respectively. As it can be clearly noted, microstructure of sample with PSR 2.0 is relatively homogeneous, and as the PSR increases to 8.1, clusters of SiC particles can be seen in the extrusion direction. These microstructures have been utilized to study and quantify different levels of clustering in materials with same reinforcement particles and matrix alloy composition. In the present work, the spatial clustering of the SiC particles is mathematically represented using two-point correlation functions and lineal path probability distributions. These data are presented in the next sections. The correlations of the spatial extent of spatial clustering of SiC particles with the process parameter PSR are studied via simulations of *realistic* microstructures, which is discussed in the next chapter.

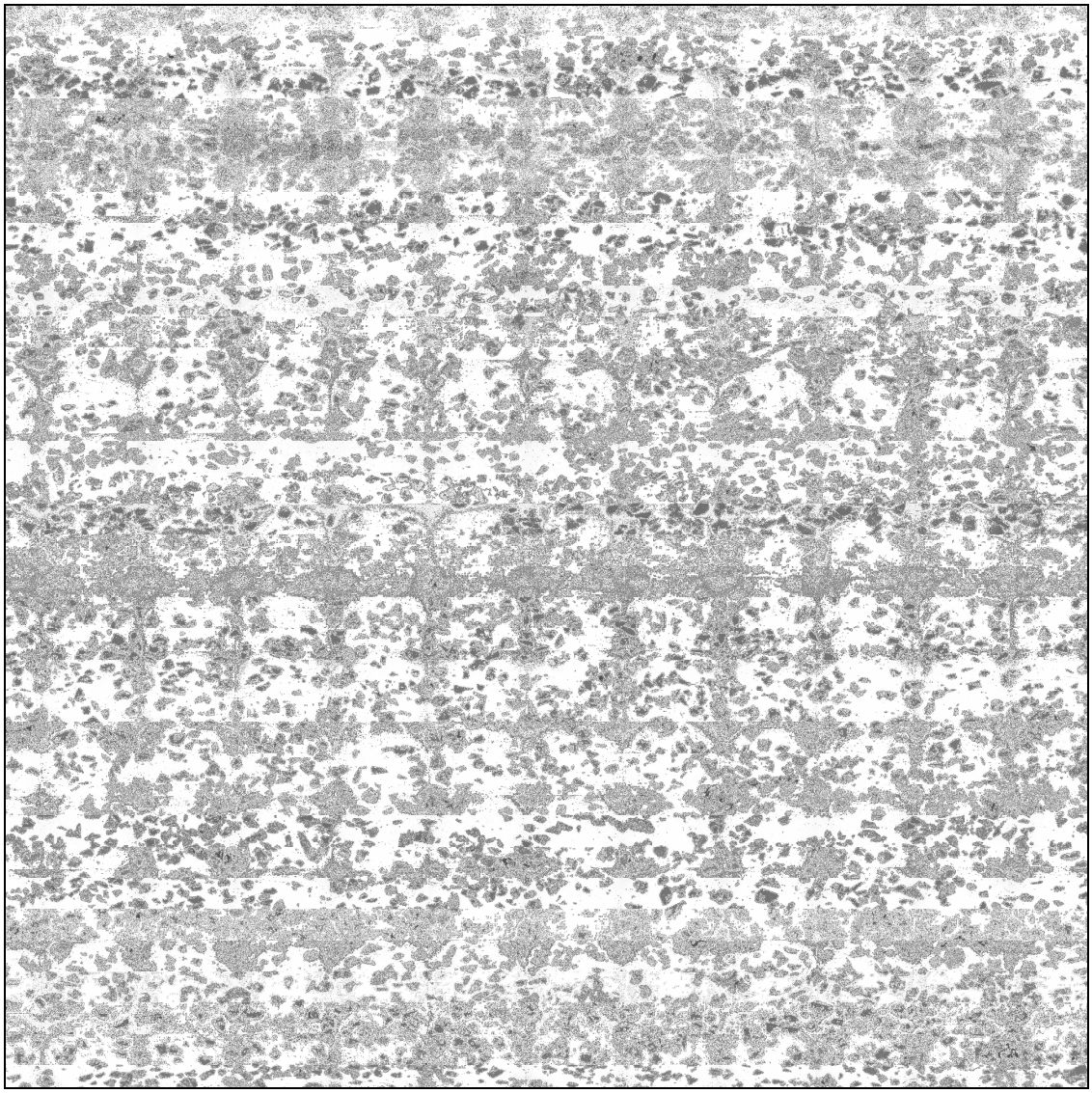



(a)

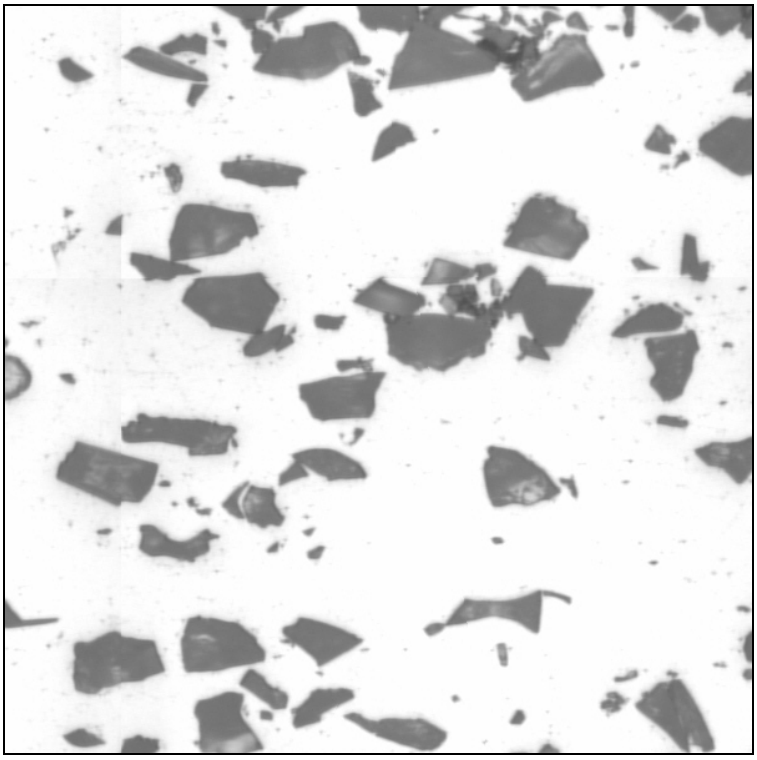



(b)

Figure 4.1: (a) Micrograph for DRA composites with PSR 2.0. (b) High resolution image showing the morphology of SiC particles.

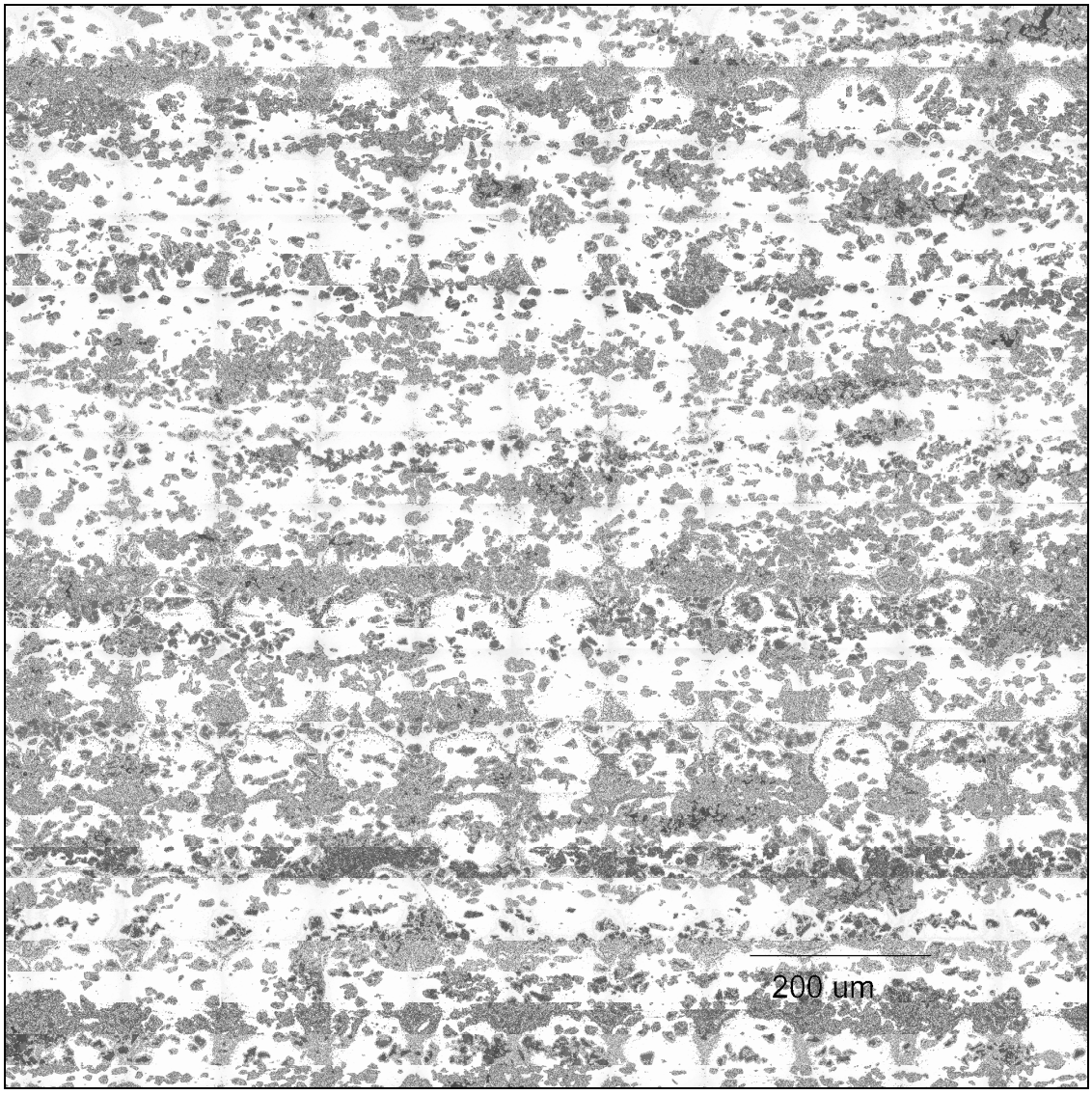


(a)

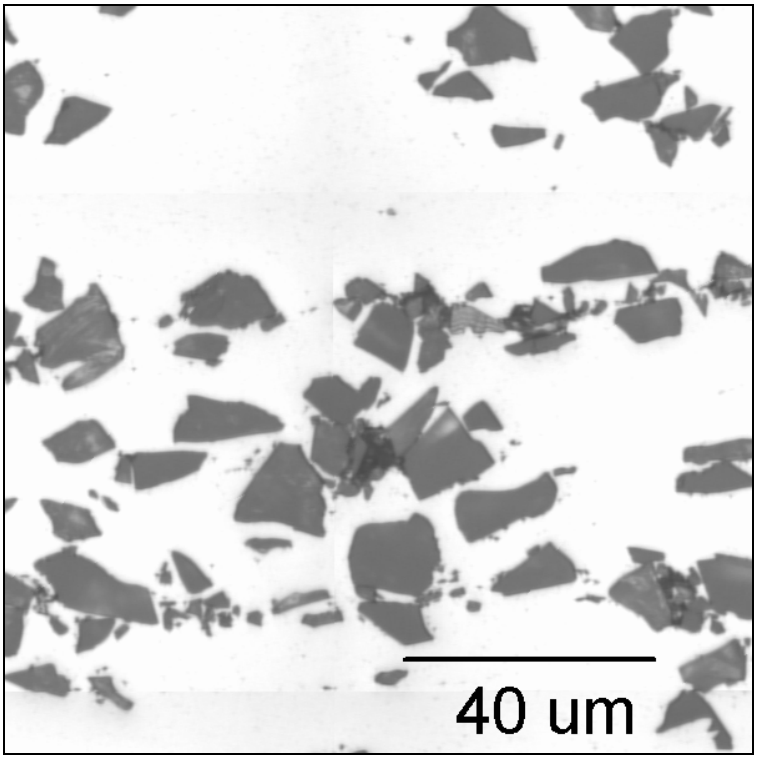


(b)

Figure 4.2: (a) Micrograph for DRA composites with PSR 3.1 (b) High resolution image showing the morphology of SiC particles.




(a)




(b)

Figure 4.3: (a) Micrograph for DRA composites with PSR 8.1 (b) High resolution image showing the morphology of SiC particles.

4.1.2 Boron Modified Titanium Alloys

The SiC particles in the DRA composites have equiaxed morphologies and have isotropic orientations. On the other hand, in numerous material microstructures the features of interest have preferred morphological orientations that lead to anisotropy in such materials. Therefore, it is of interest to account for such morphological anisotropies in the microstructure simulations. In the present work, methodology for simulations partially anisotropic microstructures is developed through its application to TiB whiskers in boron modified Ti-alloys. As mentioned in the last chapter, two types of specimens of this alloy, namely, compacted and extruded, have been used for this work. Figure 4.4 and 4.5 show the microstructures of the compacted and extruded specimens, respectively. As

can be seen from these micrographs, the TiB whiskers are non-equiaxed and hence, more susceptible to change in the morphological orientation during processing such as extrusion. Figure 4.4 depicts the microstructure of compacted boron modified titanium alloy and figure 4.5 shows the microstructure of the extruded sample. As can be noticed, the TiB whiskers in the compacted specimen are randomly oriented whereas those in the extruded specimen are aligned in the direction of extrusion. Experimental observations indicate that particle cracking is a strong function of particle orientations with respect to the loading direction, particularly when the particles have non-equiaxed shapes [99-101]. Therefore, it is of particular interest to study the relationship between processing parameters and orientations of reinforced particles. Accordingly, a new methodology is proposed in the next chapter for simulations of series of *realistic* microstructures that can mimic corresponding real microstructures arising from changes in the deformation processing parameters such as temperature and degree of plastic deformation.

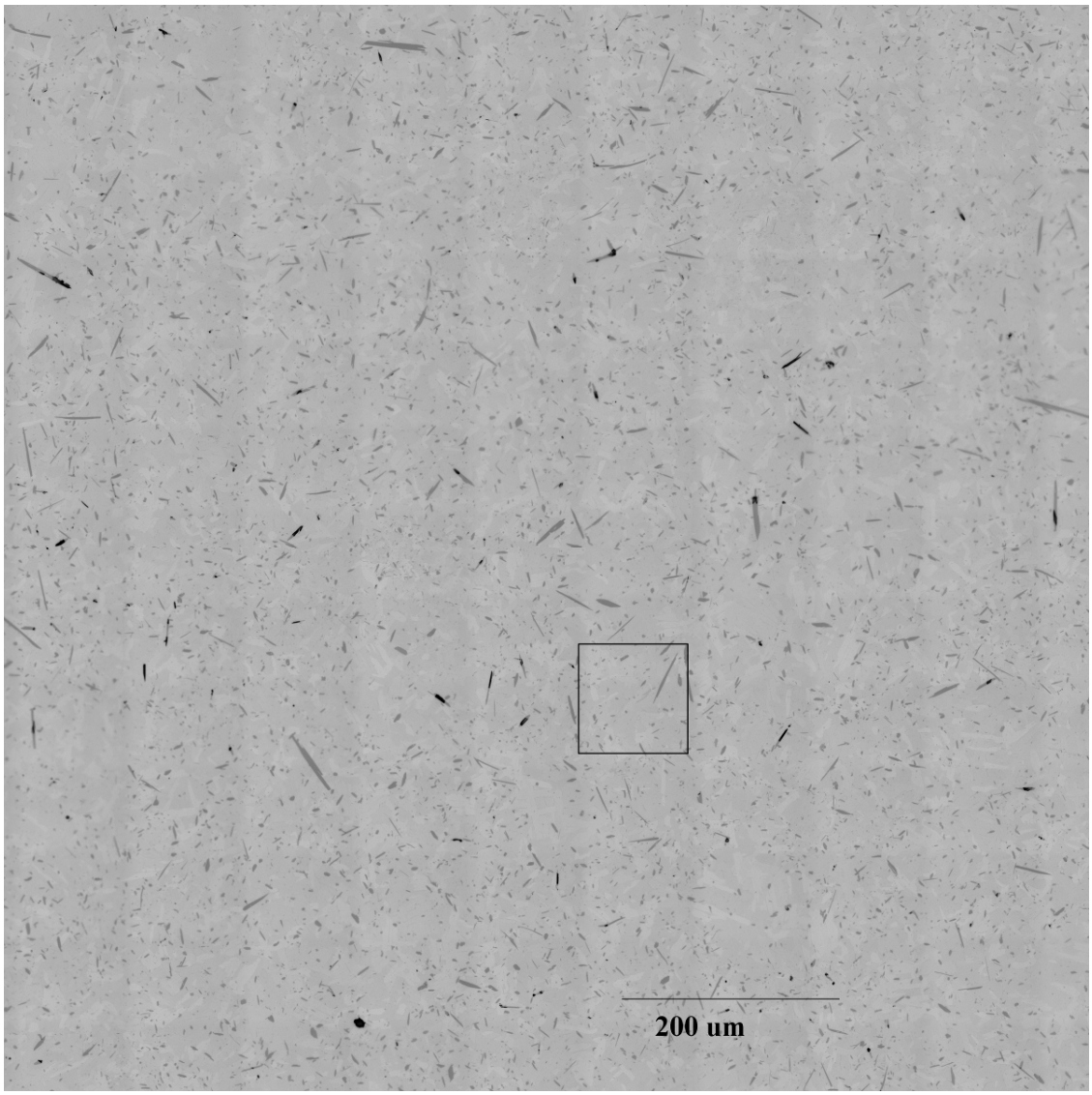



Figure 4.4: (a) Montage of the microstructure of compacted boron modified Ti-6Al-4V alloy showing randomly oriented eutectic TiB whiskers. This micrograph is a montage of 195 fields of view covering an area of approximately 1.75 $mm^2$, and has been digitally compressed for presentation.

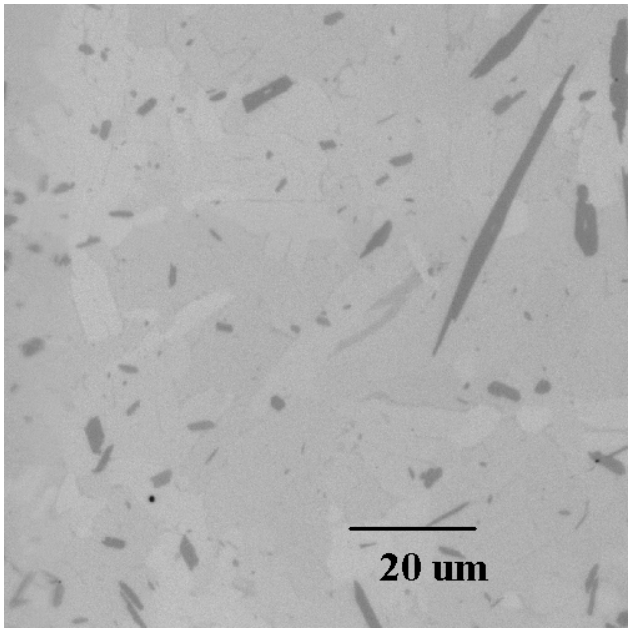



Figure 4.4: (b) Microstructure of compacted boron modified Ti-6Al-4V alloy showing randomly oriented eutectic TiB whiskers.

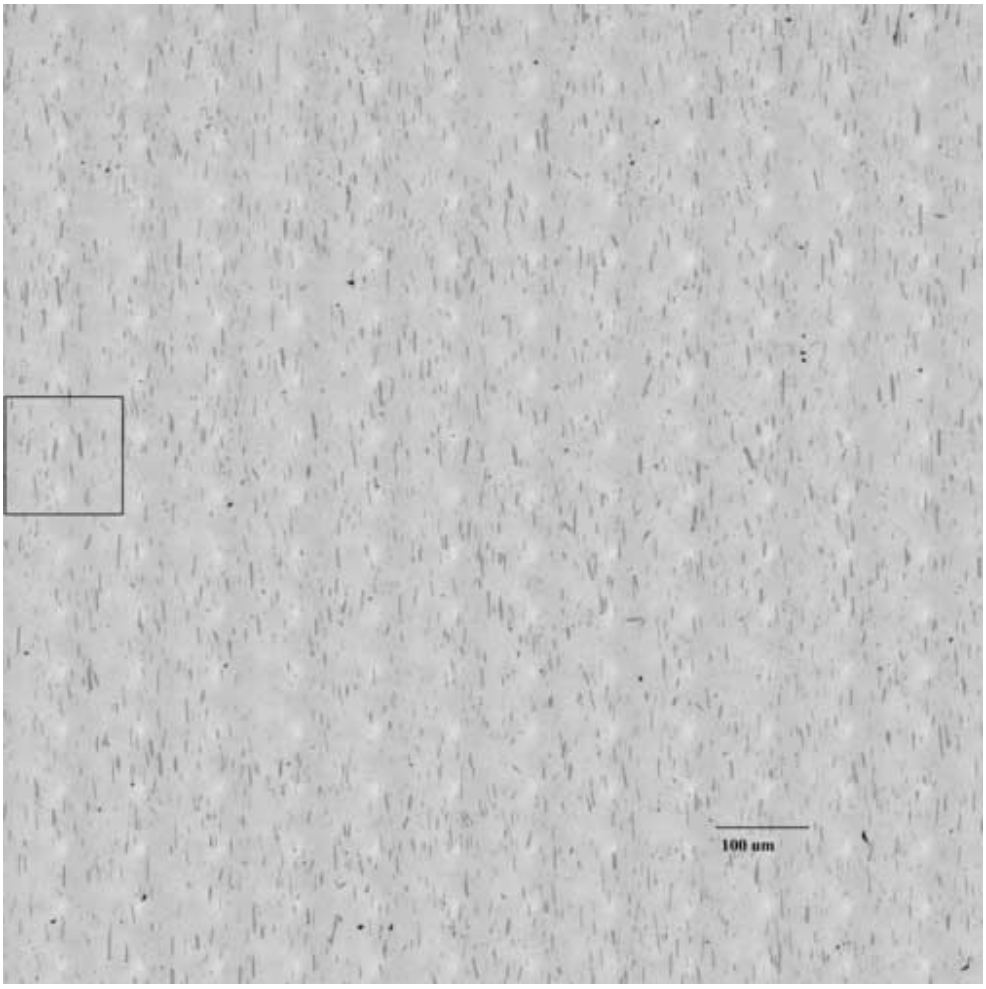



Figure 4.5: (a) Montage of the microstructure of extruded boron modified Ti-6Al-4V alloy. The extrusion direction is normal to the micrograph.

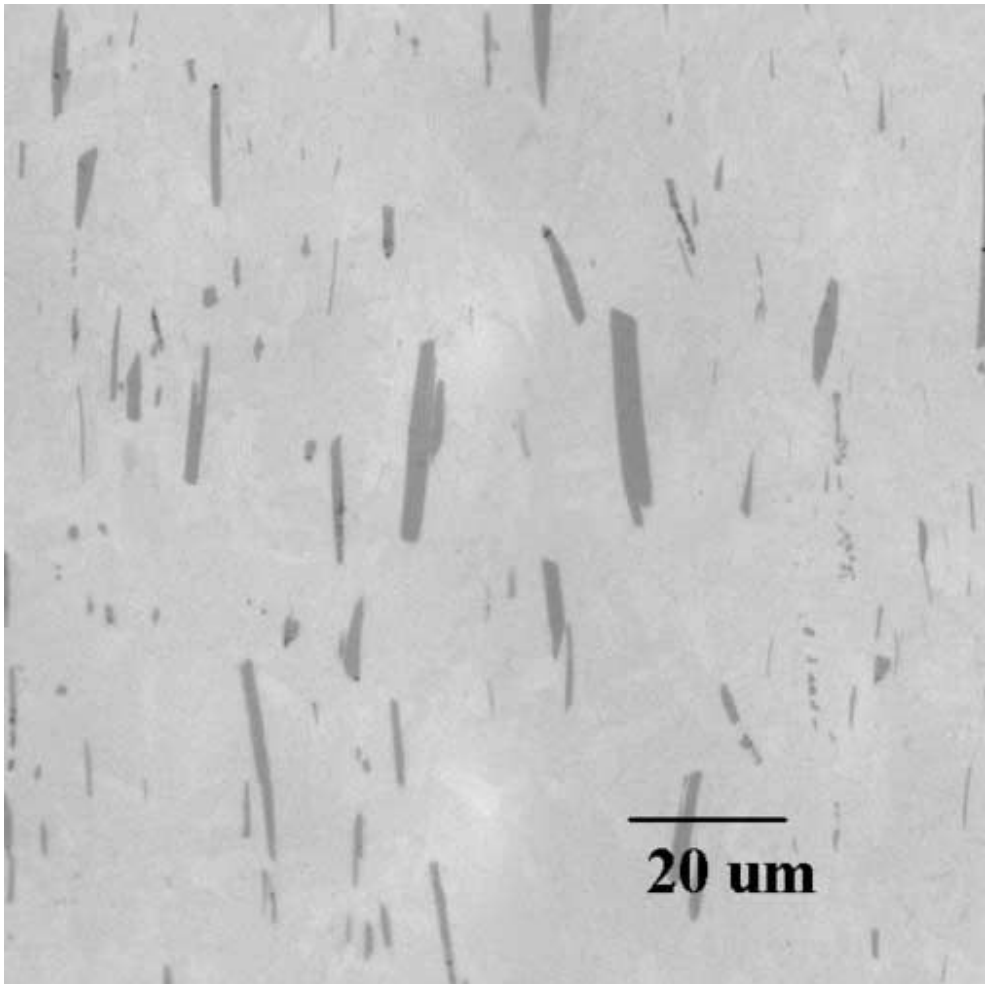



Figure 4.5: (b) Microstructure of extruded boron modified Ti-6Al-4V alloy showing TiB whiskers aligned in the direction of extrusion.

## 4.2 Stereology Based Quantitative Microstrucutral Data

### 4.2.1 DRA Composites

Conventional two-dimensional stereology techniques were used to calculate the microstructural data for the DRA composite specimens. The volume fraction of the SiC particles in the DRA composite with PSR 2.0 is 28.4%; PSR 3.1 is 28.1% and PSR 8.1 is 27.0%. Figure 4.6 shows the size distribution of the SiC particles in the DRA composites and the mean size of these particles is 7.4 μm. These calculations were performed with a KS-400 image analysis system from Kontron.

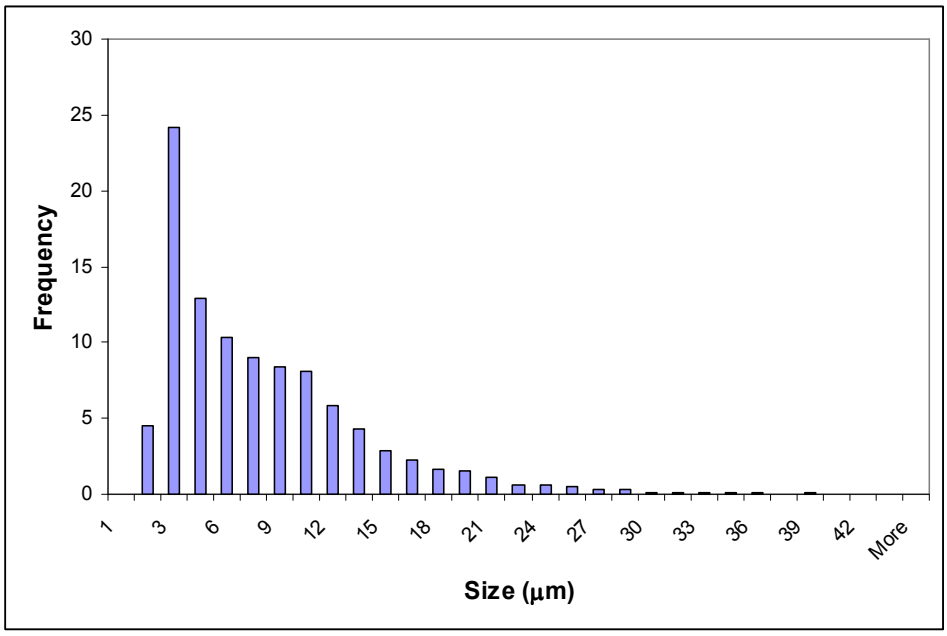



Figure 4.6: Size distribution of SiC particles in the DRA composites.

### 4.2.2 Boron modified Titanium Alloys

Image analysis system, KS-400, was used to calculate the microstructural data for the compacted and extruded boron modified Ti-6Al-4V alloy samples. Since, the extruded samples have TiB whiskers aligned in the direction of extrusion, measurements

were made on montages in the plane parallel as well as on montages in the plane normal to the extrusion direction. Average values are shown in Table 4.1 and the overall volume fraction of TiB whiskers in the compacted specimen is 4.35% and in the extruded specimen is 3.98%. Figure 4.7 shows the size-orientation distribution and Figure 4.8 shows the size-shape (ferret ratio) distribution of the TiB whiskers in the extruded boron modified Ti-6Al-4V alloy specimens.

Table 4.1 Microstructural data of boron modified Ti-6Al-4V alloy samples calculated using standard two-dimensional stereology techniques [32].

| sample | avg length (μm) | avg width (μm) | aspect ratio | volume fraction |
|---|---|---|---|---|
| compacted | 21.13 | 3.20 | 6.60 | 4.35 |
| extruded | 23.41 | 3.37 | 6.95 | 3.98 |

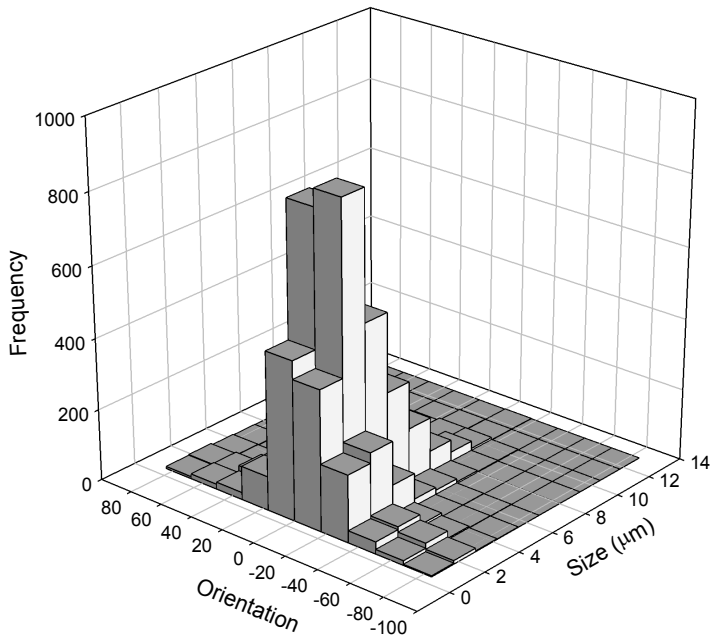



Figure 4.7: Size-orientation distribution of TiB whiskers in the extruded boron modified Ti alloys.

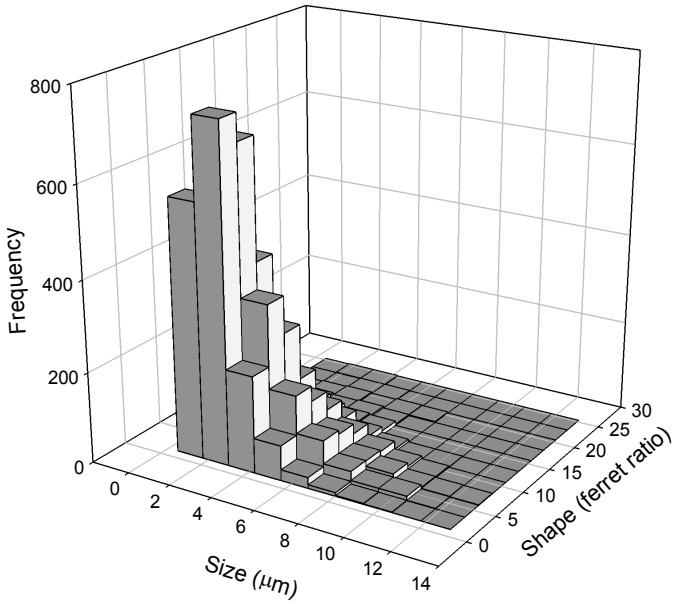


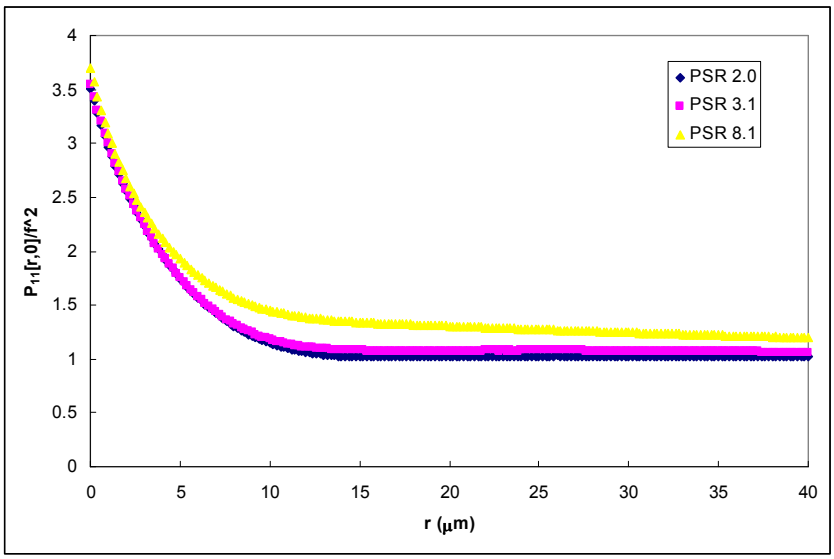

Figure 4.8: Size-shape distribution of TiB whiskers in the boron modified Ti alloys.

### 4.3 Statistical Descriptors of Microstructures

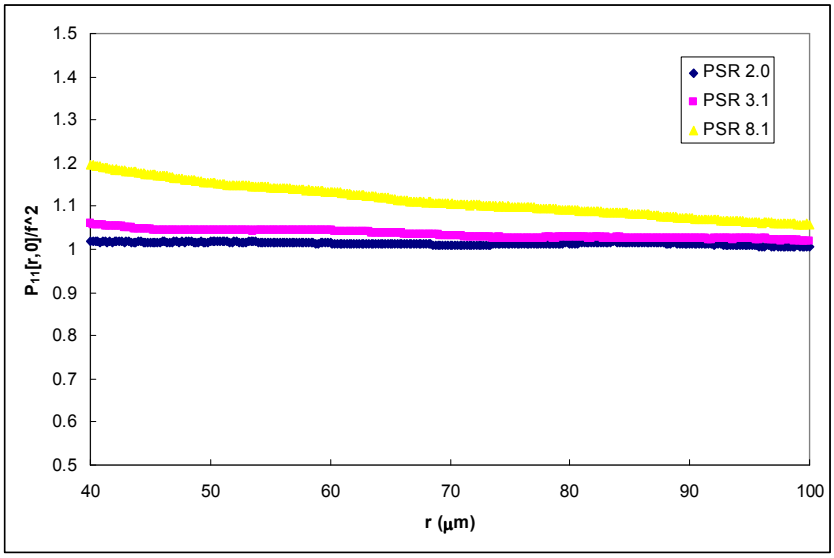
Statistical descriptors are useful for characterization of spatial arrangement and heterogeneity of microstructural features. As mentioned earlier, two-point probability functions and lineal path distributions are used in this work to characterize the microstructures of the composites using 2D montages. These functions have also been utilized in comparing the corresponding real and simulated microstructures and developing the methodology of simulation based 'realistic' microstructure, the details for which are given in the next chapter. The following subsections describe the quantitative characterization of three-dimensional microstructures using the aforementioned correlation functions.

4.3.1 Two-Point Probability Functions

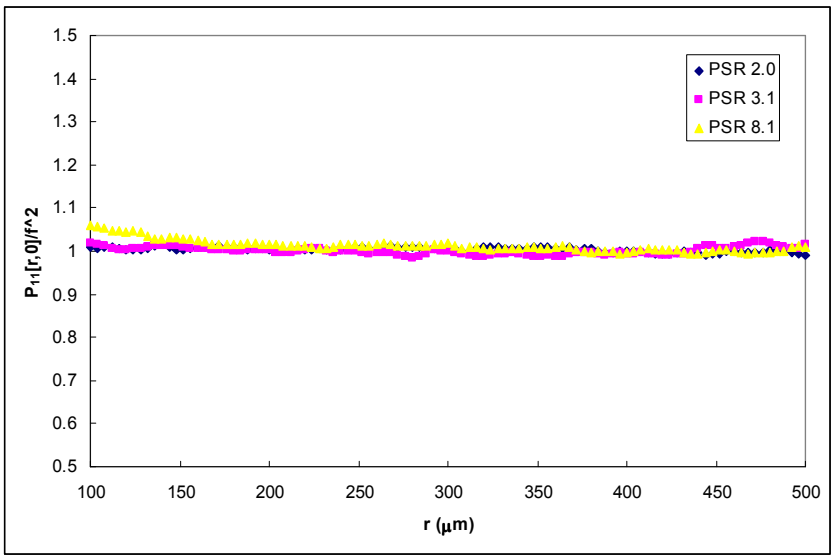
As mentioned earlier, the technique to compute two-point probability functions has been recently developed at Georgia Tech [102]. In the present work, the technique has been applied for estimation of the two-point correlation functions of the DRA composites and boron modified Ti-alloys. These functions have been utilized in capturing the spatial heterogeneities and the morphological anisotropies in the materials of interest. Figure 4.9 shows the comparison of two-point probability functions measured in the extrusion direction ($\theta = 0$) for microstructures with PSR 2.0, 3.1 and 8.1. The data has been normalized by the square of volume fraction of SiC particles, $f^2$; therefore the curve approaches unity as r goes to infinity. As expected the two-point probability function approaches unity at lowest r for the DRA composite with PSR 2.0, which is the most homogeneous of the present DRA composites and as the clustering of SiC particles is increased from 2.0 to 3.1 to 8.1, the curve takes larger values of r to approach one. Similar behavior is observed in the two-point probability functions measured in the transverse direction ($\theta = \pi/2$) for the three DRA composites as shown in Figure 4.10.


PSR 2.0
PSR 3.1
PSR 8.1
P11[r,0]/f^2
r (μm)
0
0.5
1
1.5
2
2.5
3
3.5
4
0
5
10
15
20
25
30
35
40

(a)

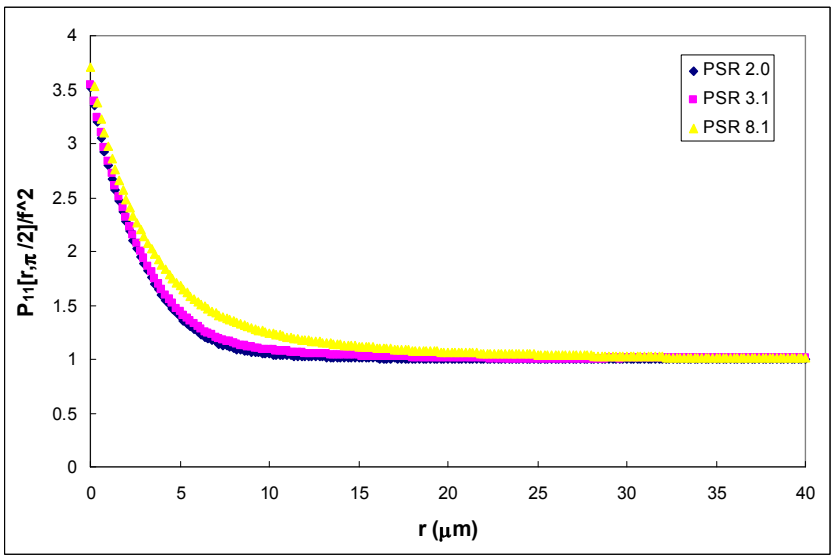

PSR 2.0
PSR 3.1
PSR 8.1
P11[r,0]/f^2
r (μm)
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5
40
50
60
70
80
90
100

(b)

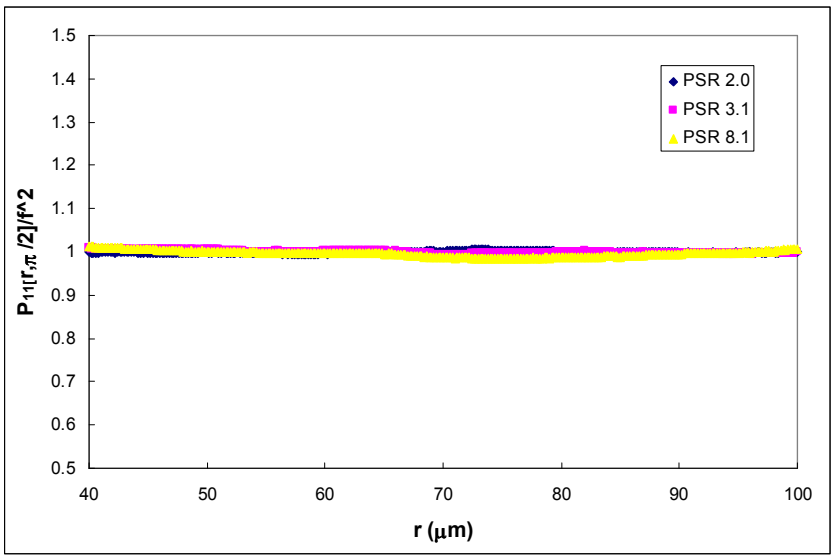



(c)

Figure 4.9: Two-point probability function comparison for DRA composites with PSR 2.0, 3.1 and 8.1 measured in extrusion direction. a) short-range, b) intermediate-range and c) long –range.




(a)




(b)

Figure 4.10: Two-point probability function comparison for DRA composites with PSR 2.0, 3.1 and 8.1 measured in transverse direction. a) Short-range; b) intermediate-range.

Two-point probability functions have also been used to compare the morphological anisotropies in the microstructure of boron modified Ti-alloys containing TiB whiskers. As shown earlier, these alloys can have different morphological orientations of TiB whiskers depending upon the processing routes. Comparison of two-point probability functions measured in the microstructure of extruded boron modified Ti-alloys in three different directions, extrusion direction ($\theta = 0$), transverse direction ($\theta = \pi/2$) and at an angle of $45^{o}$ between extrusion and transverse direction is shown in Figure 4.11. The two-point probability function in the extrusion direction has highest data values for each r before it stabilizes at the square of the volume fraction of the TiB whiskers, which is expected since the TiB whisker are aligned in the extrusion direction. Whereas, the compacted boron modified Ti-alloys contains non-aligned TiB whiskers and hence an isotropic microstructure, which is observed in the two-point probability functions

measured for their microstructure in three different directions as shown in Figure 4.12. As expected, the curves follow similar path for all the two-point probability functions measured in different directions for the microstructure of compacted boron modified Ti-alloy. These experimental data for the real microstructures have been used to compare with the corresponding data for simulated microstructures to ensure that the simulated microstructures represent the corresponding microstructural reality, the details of these comparisons is presented in the following Chapters.

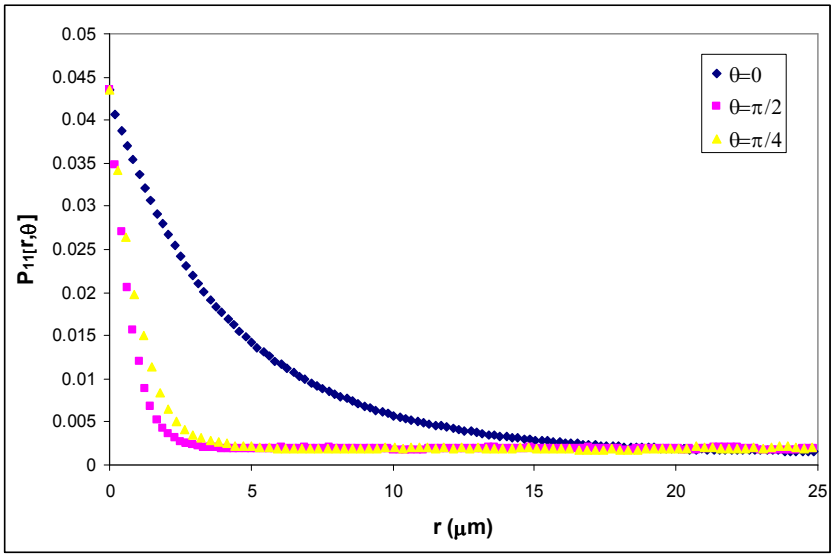



Figure 4.11: Comparison of two-point probability functions for the extruded boron modified Ti-alloy in extrusion direction ($\theta = 0$), transverse direction ($\theta = \pi/2$) and intermediate direction ($\theta = \pi/4$).

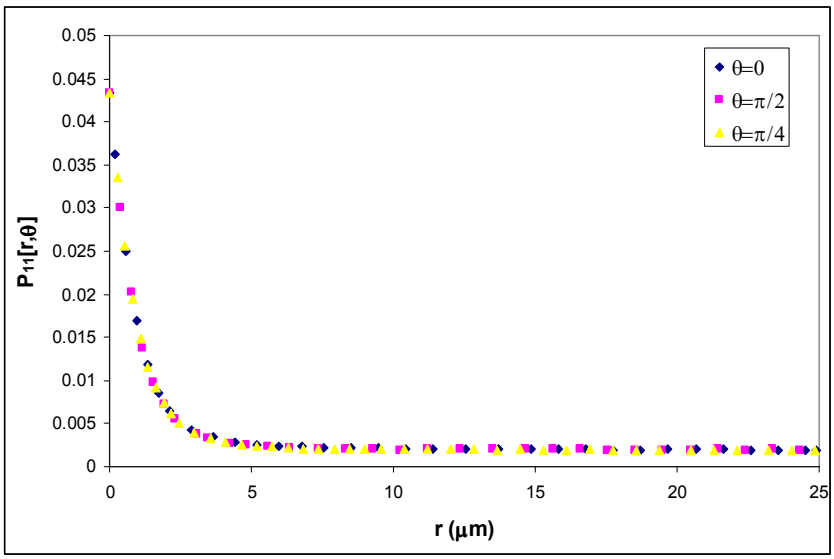



Figure 4.12: Comparison of two-point probability functions for the extruded boron modified Ti-alloy in extrusion direction ($\theta = 0$), transverse direction ($\theta = \pi/2$) and intermediate direction ($\theta = \pi/4$).

4.3.2 Lineal Path Probability Distributions

Computations of lineal path probability distributions are required for representation of short range, *intermediate range, and long-range* spatial patterns, correlations, anisotropies, and heterogeneities in microstructures. In the present work, a computer code has been developed to compute the lineal path probability distribution functions in a flexible (for any given direction) and efficient (less time consuming) manner. The code has been applied to the microstructure of boron modified Ti-alloys to compute the three-dimensional lineal path distributions from 2-D vertical sections. Figure 3a shows digitally compressed montage of microstructure of the blind die compacted and extruded boron modified Ti-6Al-4V alloy depicting anisotropic TiB whiskers along the extrusion plane. The montage consists of 195 contiguous fields of view. Each field of view of the montage has been grabbed at high resolution depicted in Figure 3b. This

microstructure represents the extrusion plane containing the extrusion axis. If extrusion axis has to be the symmetry axis then the 2D microstructure in the transverse plane (which is perpendicular to the extrusion axis) must be isotropic in that plane. The symmetry around the extrusion axis can be verified by comparing lineal path functions in different directions on the transverse plane. Figure 4.13 shows the comparison of lineal path function in three different directions as measured on the transverse plane (figure 4.14). Since they all follow the similar curve it can be concluded that for this specimen the extrusion axis is the axis of symmetry and any given extrusion plane must represent the vertical section irrespective of its orientation.

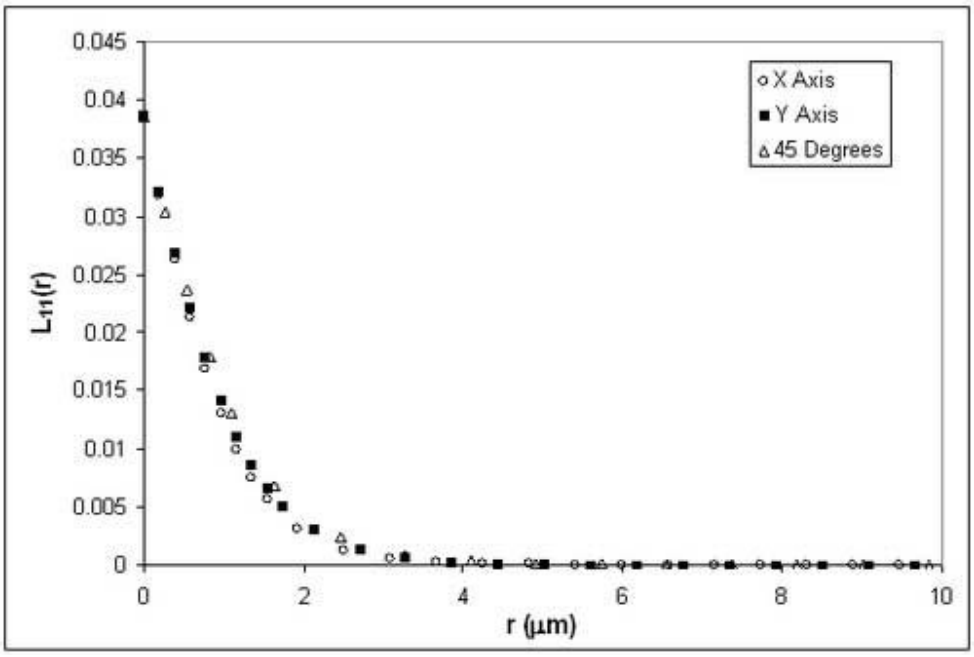
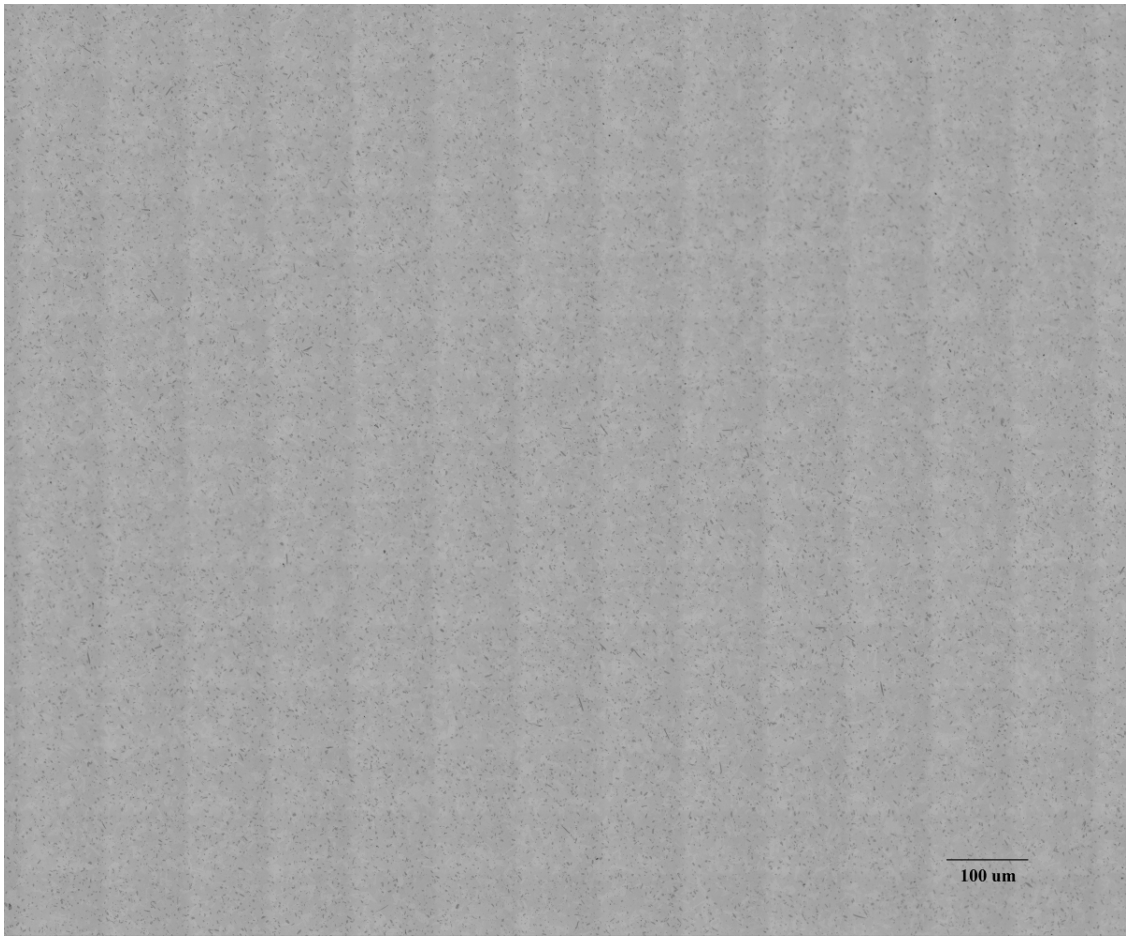
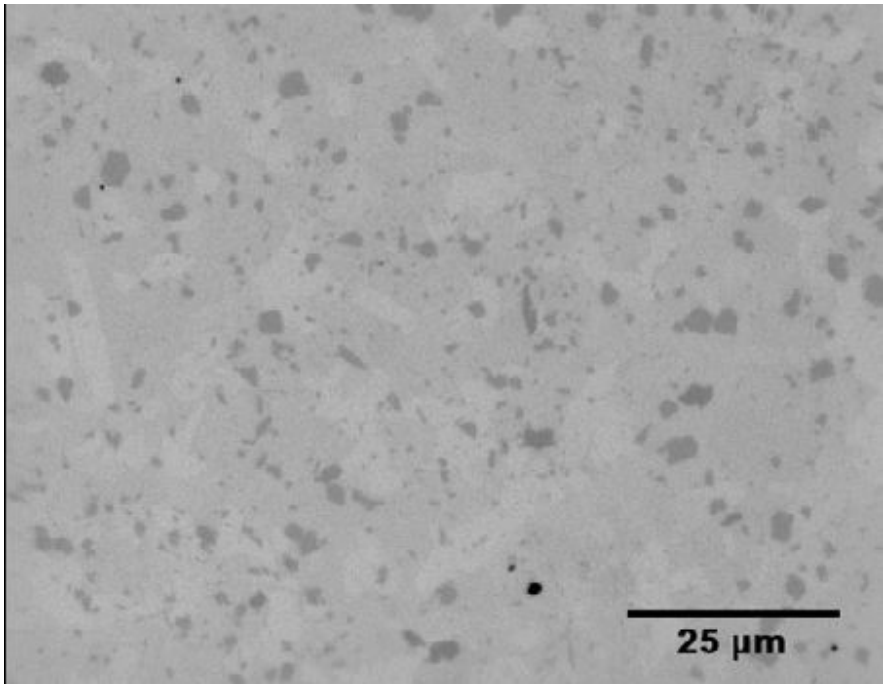



Figure 4.13: Comparison of particle lineal path functions in three different directions as measured on the transverse plane of extruded boron modified Ti-6Al-4V alloy.




(a)




(b)

Figure 4.14(a): Microstructure in the transverse plane for extruded boron modified Ti-6Al-4V alloy. (b) High resolution image showing the cross-section of the TiB whiskers.

Figure 4.15a shows the lineal path probability function $L_{11}$ (r, θ) for the lineal paths in the TiB whiskers in the anisotropic microstructure of Figure 4.5a along the extrusion direction (Y-direction, $\theta = 0$), transverse direction (X-direction, $\theta = 90$ degrees), and along the direction at an angle of 45 degrees with respect to the extrusion direction. On the other hand, Figure 4.15b shows the lineal path probabilities $L_{22}$ (r, θ) for the *paths in the matrix* along the extrusion direction ($\theta = 0$), transverse direction ($\theta = 90$ degrees), and along the direction at an angle of 45 degrees with respect to the extrusion direction (i.e. $\theta = 45$ degrees). As expected, for this anisotropic microstructure, $L_{11}$ (r, θ) and $L_{22}$ (r, θ) vary significantly with the angular orientation of the lineal path, θ. It can be seen in Figure 4.15a that for the lineal paths in the TiB

whiskers along transverse direction approaches zero rapidly as compared to that along the extrusion direction. This is because of the fact that the particles are elongated in the extrusion direction for this microstructure. It may also be noted that length scales of a microstructure can be analyzed by looking at the corresponding lineal path functions in different directions and comparing the point where the curve approaches zero. Figure 4.15b depicts that the probability of the lineal path in the matrix $L_{22}$ (r, θ) along the extrusion direction approaches zero only at very large distances (~ 150 μm), whereas that along the transverse direction approaches zero at relatively short distances (~ 35 μm). This is again due to microstructural anisotropy. On the other hand, the lineal path probabilities in the TiB whiskers approach zero at short distances that represent the length scales of the TiB whiskers along different directions.

Figure 4.4a shows digitally compressed montage of microstructure of the blind die compacted boron modified Ti-6Al-4V alloy depicting *isotropic random* TiB whiskers, which has been created by pasting together contiguous microstructural fields having high resolution depicted in Figure 4.4b. Figures 4.16a and 4.16b depict the lineal path probabilities for this isotropic microstructure for the lineal paths in the TiB whiskers and in the matrix, respectively. As expected, in this case, the lineal path probabilities do not vary with the orientation angle of the path. Further, the lineal paths in the matrix approach zero at about 50 μm or so, which is significantly lower than that along the extrusion direction in the anisotropic microstructure (see Figure 4.15a).

Two-point correlation functions are often used for microstructure representation. Nonetheless, the two-point correlation functions represent the probabilities for the events that only concern the *end points* of the line, whereas the lineal path probabilities are for

the events that concern the complete line segment length. Therefore, in general, these two probabilities are not equal except at very small distances. Figure 4.19a depicts lineal path probability for the paths in the matrix, $L_{22}$ (r, θ), and the two-point correlation function $P_{22}$ (r, θ) along the extrusion direction for the anisotropic microstructure shown in Figure 43 and figure 4.19b compares these functions in transverse direction. In this case, $P_{22}$ (r, θ) is the probability that both the end points of a randomly located line of length r that is parallel to the extrusion direction are in the matrix. Figure 4.19 clearly demonstrates that the probabilities $L_{22}$ (r, θ) and $P_{22}$ (r, θ) are quite different and they are approximately equal only at very short distances. Therefore, the lineal path probability distributions contain the information that is not reflected in the two-point correlation functions. However, as both of these probabilities can be computed from the same digital images, it may be useful to utilize both descriptors for microstructure representation. Using both probability functions for simulations of realistic microstructures is expected to lead to simulated microstructures whose geometry is closer to the corresponding real microstructures that the ones simulated using just two-point correlation functions.

The algorithm discussed in this contribution provides the technique for measuring lineal path functions in 0, 90 and 45 degree angles. The lineal path function for any other particular direction can be measured by rotating the image by desired angle and then using this algorithm on a largest possible rectangular portion cut out from the rotated image. Since the original image is large enough the rectangular area edited out from the rotated image provides sufficient pixel data points to compute statistically accurate lineal path functions. One such example is shown in figure 13 where lineal path function along

7 degrees from the extrusion axis for the image shown in 4.5a is plotted. As expected, this curve lies between the lineal path functions for extrusion and transverse direction.

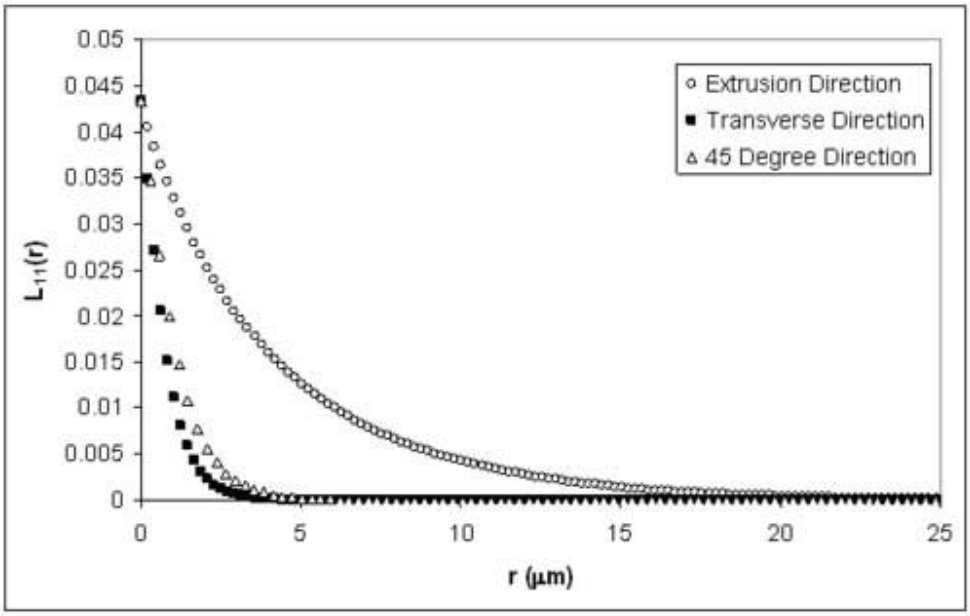



Figure 4.15a: Lineal path probability function for the paths in the TiB whiskers for the anisotropic microstructure in Figure 4.3.

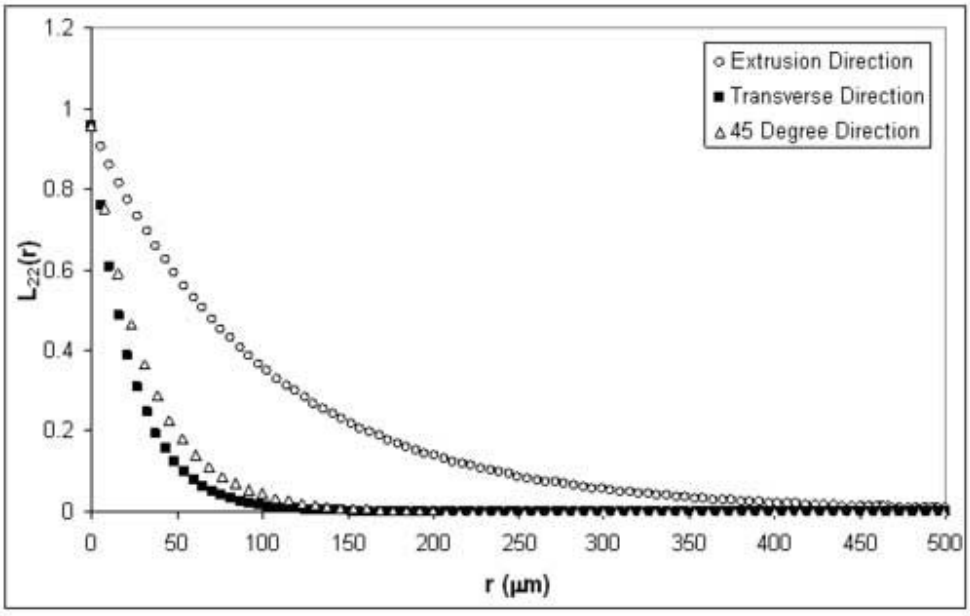



Figure 4.15b: Lineal path probability function for the paths in the matrix for the anisotropic microstructure in Figure 4.5.

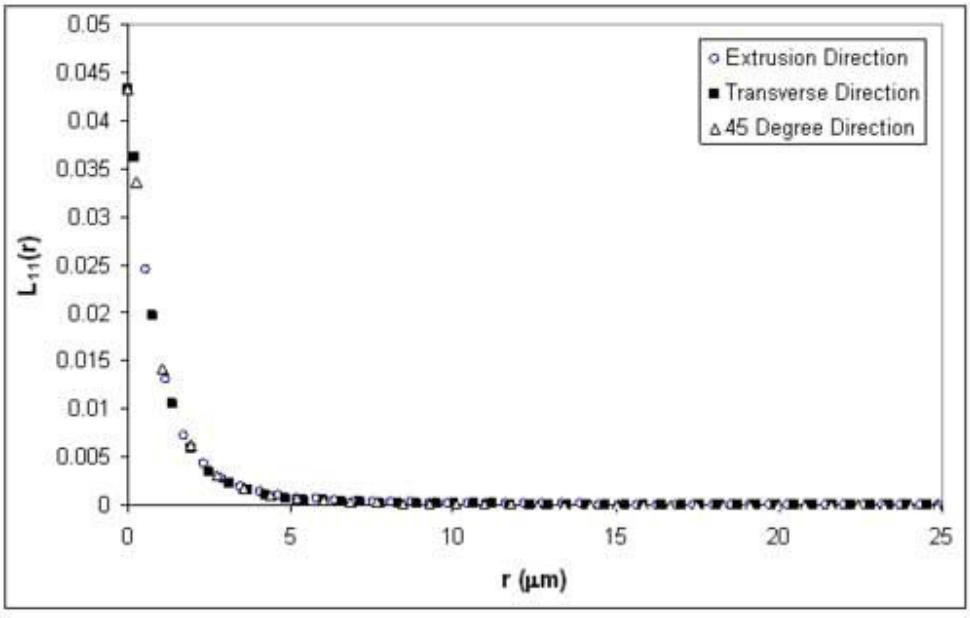



Figure 4.16a: Comparison of Lineal Path Functions in different directions for the TiB whiskers in the isotropic random microstructure of Figure 4.2.

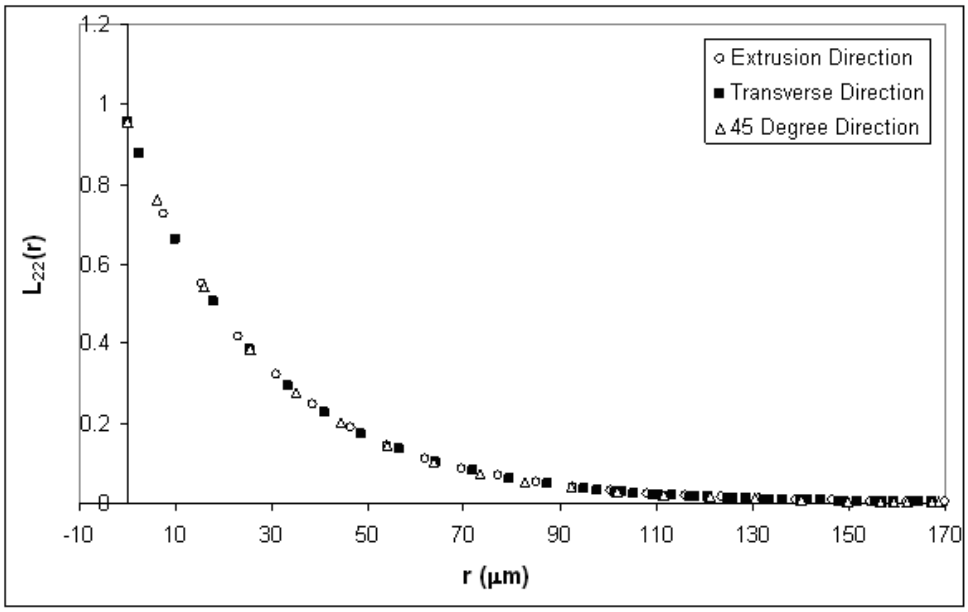



Figure 4.16b: Comparison of Lineal Path Functions in different directions for the matrix in the isotropic random microstructure of Figure 4.4.

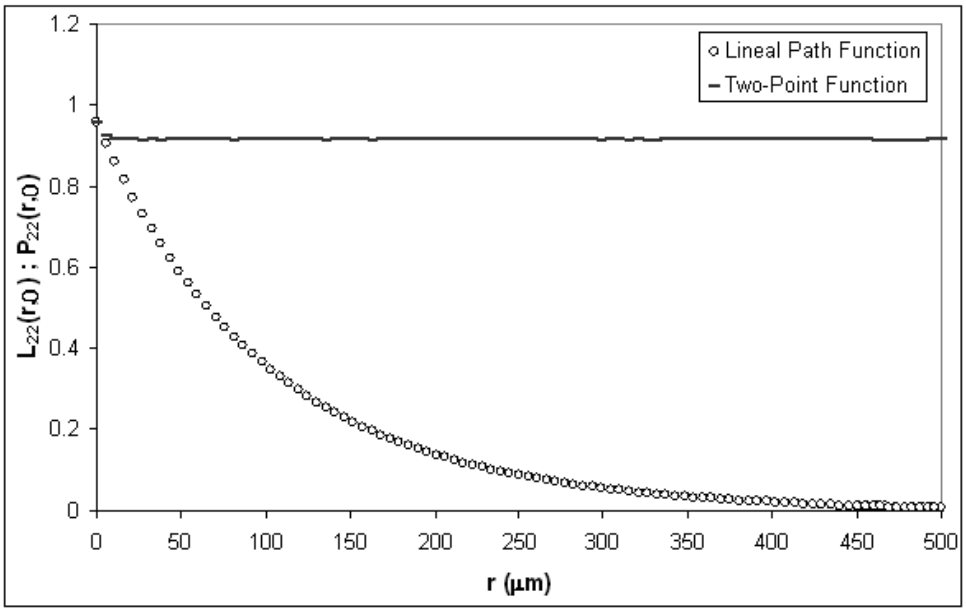



Figure 4.17a: Comparison of Lineal Path Functions and Two-point function in the extrusion direction for the matrix in the anisotropic random microstructure of Figure 4.3.

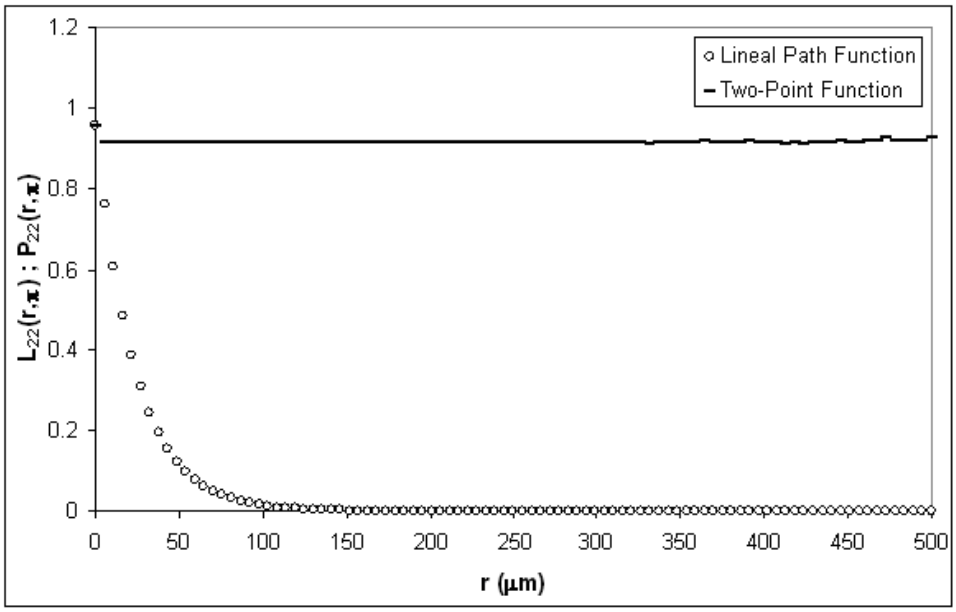



Figure 4.17b: Comparison of Lineal Path Functions and Two-point function in the transverse direction for the matrix in the anisotropic random microstructure of Figure 4.5.

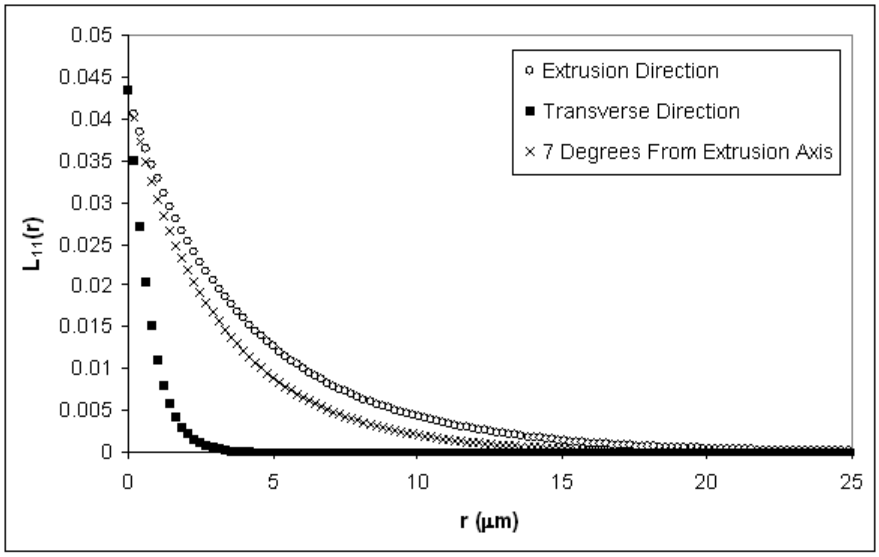



Figure 4.18: Lineal path probability functions for the paths in the TiB whiskers for the anisotropic microstructure in Figure 4.5.

## 4.4 Visualization of 3D microstructures

The reconstruction of three-dimensional microstructures was done by using the montage-serial sectioning technique. The details of this technique are described in the previous chapter. The following subsections detail the qualitative analysis done by visualization of 3D microstructures for the composites of interest.

### 4.4.1 Al-$SiC_P$ Composites

In the present work, 3D microstructural visualization has been done using 100 montage serial sections; each montage serial section containing 100 contiguous microstructural fields grabbed at 500×. Therefore, the resulting 3D data sets are useful for characterization and visualization of short-, intermediate- and long-range spatial clustering and connectivity of SiC reinforcement particles in the specimens of discontinuously reinforced aluminum alloy (DRA) composites.

Figure 4.21.a shows a stack of 20 aligned montage serial sections for the 8.1 PSR composite microstructure, where each serial section is a digitally compressed montage of 100 contiguous microstructural fields. The spatial clustering of SiC particles and long-range arrangement of particle-rich and particle-poor regions is revealed in this serial section stack. Focus on the bordered region in Figure 4.21a. Figure 4.21b is the magnified view of that bordered region, where each section contains about 20 contiguous microstructural fields of the montage in Figure 4.21a. The SiC particle clustering is revealed at this length scale as well. In Figure 4.21b, observe the changes in the size of the SiC particles at the edges of these serial sections as well as appearance and disappearance of the SiC particles in the successive serial sections. Figure 4.21c is a magnified view of the small-bordered region in Figure 4.21b; this is the magnification at

which all microstructural fields have been grabbed. Figure 4.21c is exactly the stack of serial sections generated by the classical serial sectioning technique [58, 103, 104]. The spatial clustering of SiC particles is not evident in Figure 4.21c. Therefore, the aligned stack of montage serial sections contain complete long-range (on the order of millimeters, see Figure 4.21a), intermediate-range (10–100 μm, Figure 4.21b) as well as short-range (1–10 μm, Figure 4.21c) microstructural information that can be used to quantify the spatial arrangement of microstructural features, formation of long-range clusters particle-rich bands, chains of connected particles/grains, etc. The stacks of aligned montage serial sections are also extremely useful for characterization of attributes such as connectivity and coordination number distribution [61], number density of features [95], and the 3D particle/grain size distributions [64].

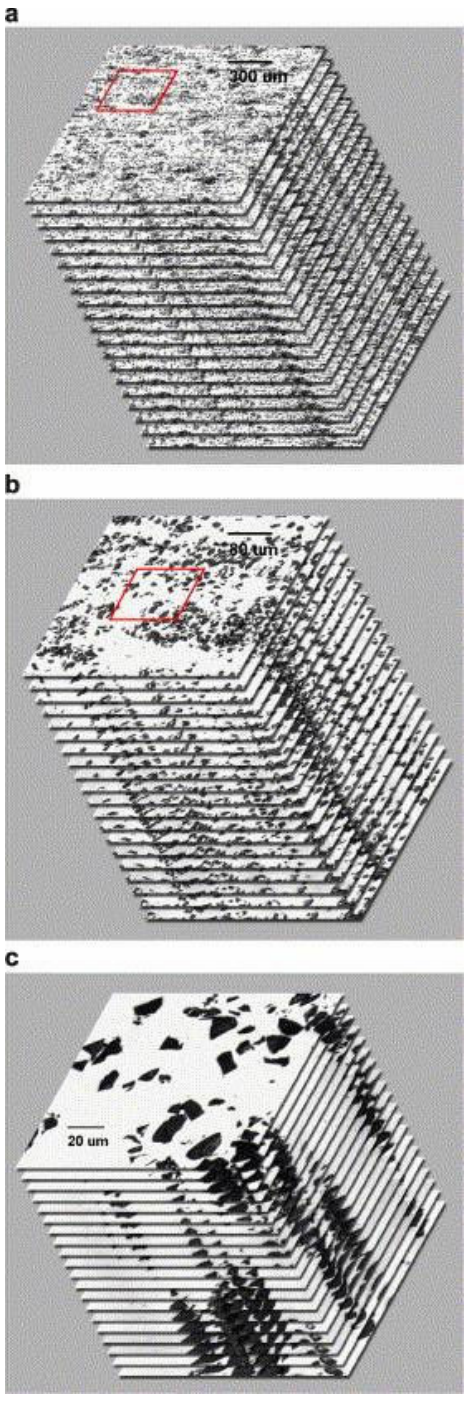



Figure 4.19: (a) Stack of 20 montage serial sections for the 8.1 PSR specimen microstructure. (b) The magnified view of the small bordered region of the stack of 20 montage serial sections for the 8.1 PSR specimen microstructure in (a). (c) The magnified view of the small bordered region of the stack of 20 montage serial sections for the 8.1 PSR specimen microstructure in (b).

Figure 4.20a shows a stack of 20 aligned montage serial sections for the 2.0 PSR composite microstructure, where each serial section is a digitally compressed montage of 100 contiguous microstructural fields. Comparison of Figure 4.21 and Figure 4.20 reveals significant differences in the spatial arrangement of SiC particles in the two microstructures. The particle-rich and particle-poor regions are not observed in Figure 4.20a, and the SiC particles are spatially uniform. Figure 4.20b is the magnified view of the bordered region in Figure 4.20a, where each section contains about 20 contiguous microstructural fields of the montage in Figure 4.20a. The SiC particles are not spatially clustered at this length scale either. Also, in Figure 4.20b, observe the changes in the size of the SiC particles at the edges of these serial sections as well as appearance and disappearance of the SiC particles in the successive serial sections. Figure 4.20c is a magnified view of the small-bordered region in Figure 4.20b; this is the magnification at which all microstructural fields have been grabbed. The spatial arrangement of SiC particles revealed in Figure 4.20c is not very different from that in Figure 4.21c, which demonstrates the need of montage serial sectioning for characterization of long range spatial clustering of features in the microstructures.

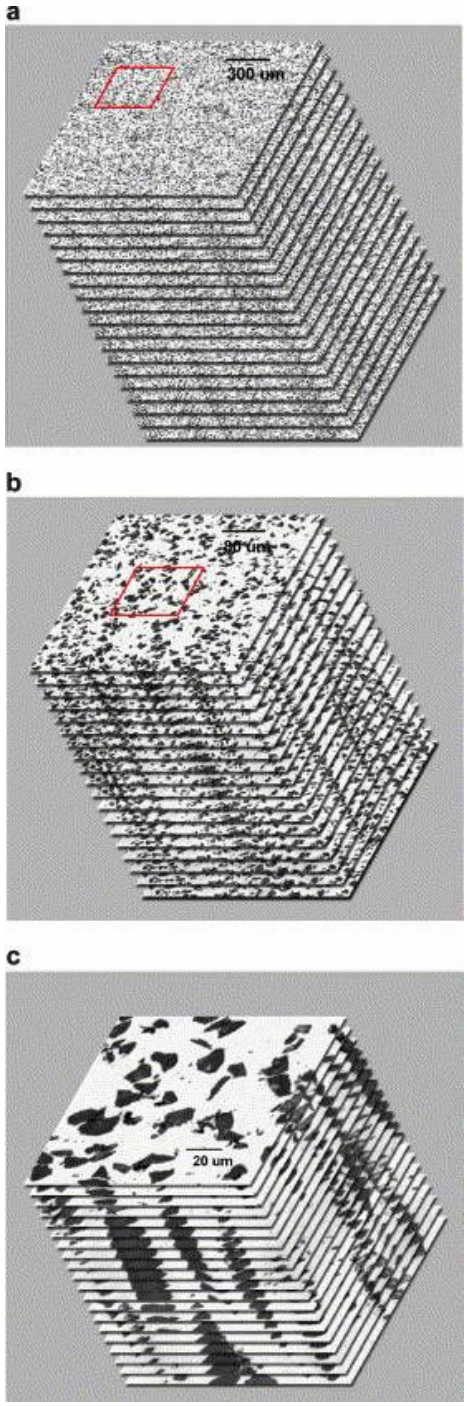



Figure 4.20: (a) Stack of 20 montage serial sections for the 2.0 PSR specimen microstructure. (b) The magnified view of the small bordered region of the stack of 20 montage serial sections for the 2.0 PSR specimen microstructure in (a). (c) The magnified view of the small bordered region of the stack of 20 montage serial sections for the 2.0 PSR specimen microstructure in (b).

Figure 4.21a shows a *small* segment of volume rendered 3D microstructure of the 8.1 PSR composite microstructure. The volume-rendered visualization is useful for implementation of the 3D microstructural images in the finite element (FE) based computations of the micro-mechanical response [105]. Figure 4.21b shows the surface rendered 3D microstructure of the 8.1 PSR DRA composite, whereas Figure 4.21c depicts another type of surface rendering of the 3D microstructure, where the SiC particles are effectively removed from the microstructure leaving behind only the matrix. Note that the 3D microstructures displayed in Figure 4.21 are only about 5% of the actual 3D microstructural volumes contained in the 100 montage serial sections. The surface rendered 3D microstructure of the 2.0 PSR composite depicted in Figure 4.22a and b, reveal an almost uniform random spatial arrangement of the SiC particles in the Al-alloy matrix: no significant particle clustering is observed. On the other hand, the microstructure of the 8.1 PSR composite shown in Figure 4.21b clearly reveals highly clustered long-range spatial arrangement of the SiC particles in the form of long particle-rich bands. The quantitative measurements reveal that these bands are on the average 280 μm long and 7800 $\mu m^2$ in the cross-sectional area perpendicular to the extrusion direction. Such information is extremely useful for modeling such non-uniform microstructures, and it can be obtained only from large-volume high-resolution 3D microstructure generated by a stack of montage serial sections: classical serial sectioning is not useful for this purpose. The high-resolution of montage serial sections and large volume permits extraction of individual bands of SiC particle-rich regions from the 3D microstructure for a more detailed study. Figure 4.23 displays one such particle-rich band

containing highly clustered SiC particles. It is also possible to examine the SiC particle clustering and contiguity in the particle-rich bands. Figure 4.24 displays one such chain of connected SiC particles in the particle-rich band in Figure 4.23. The individual particle morphologies also can be examined via extraction of individual SiC particles from the 3D images. For example, Figure 4.25 shows a few individual SiC particle extracted from the 3D microstructure. Thus, the high-resolution, large-volume 3D data set generated by montage serial sectioning permits detailed interrogation of 3D microstructure at length scales from 1 to 1000 μm (1 mm).

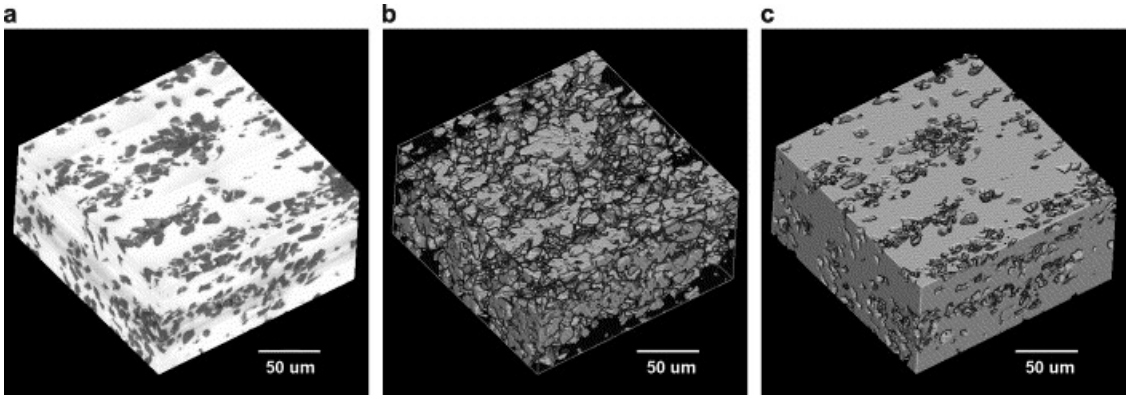



Figure 4.21: (a) Small segment of volume-rendered reconstructed 3D microstructure of the 8.1 PSR specimen. (b) Small segment of *surface*-rendered reconstructed 3D microstructure of the 8.1 PSR specimen. (c) Small segment of inverted-contrast *surface*-rendered reconstructed 3D microstructure of the 8.1 PSR specimen, where the SiC particles are removed leaving behind just the matrix.

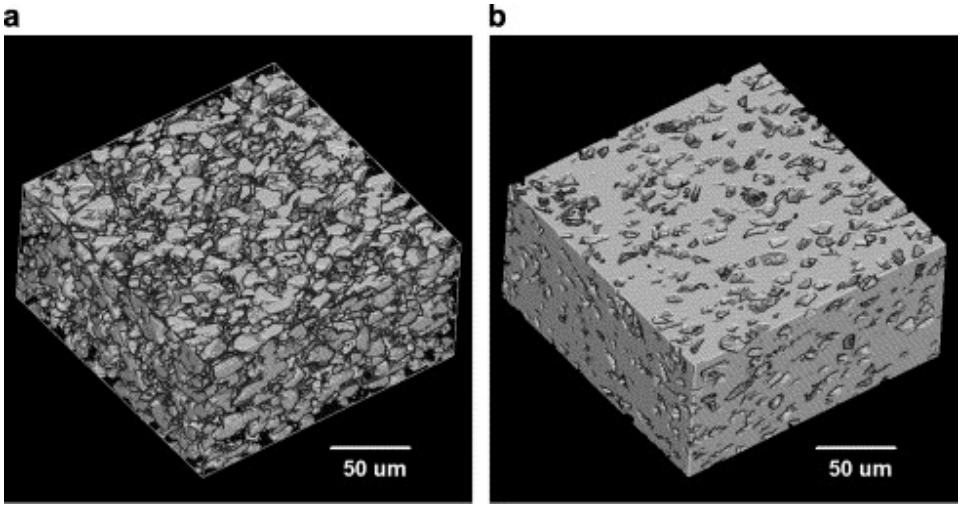



Figure 4.22: (a) Small segment of *surface*-rendered reconstructed 3D microstructure of the 2.0 PSR specimen. (b) Small segment of inverted contrast *surface*-rendered reconstructed 3D microstructure of the 2.0 PSR specimen, where the SiC particles are removed leaving behind just the matrix.

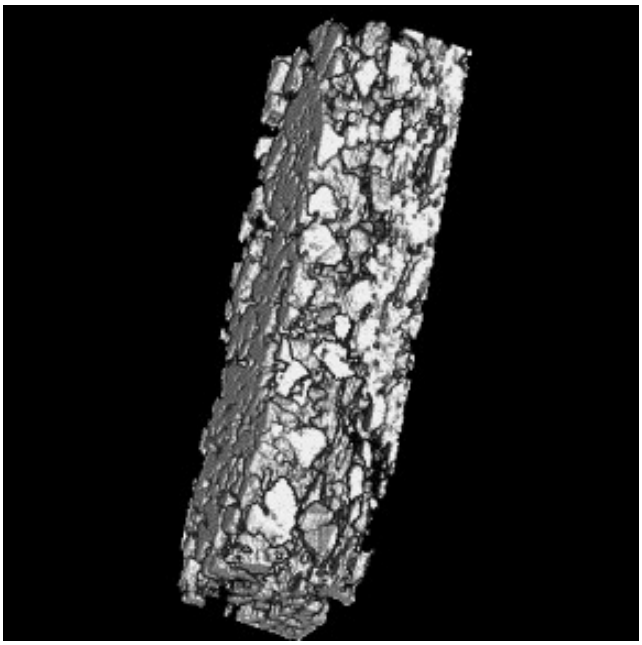


Figure 4.23: A particle-rich band containing clustered SiC particles in the 8.1 PR microstructure.

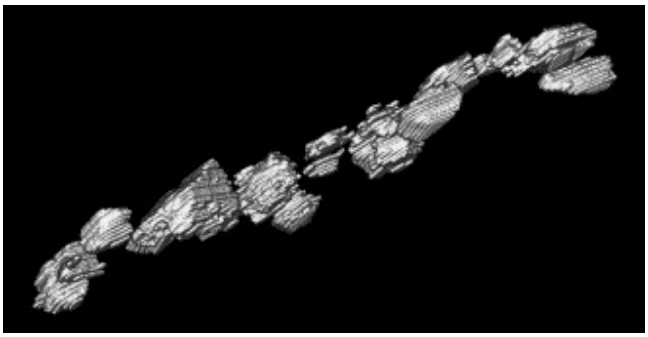


Figure 4.24: A chain of connected SiC particles in the 8.1 PSR microstructure. The particle-rich bands (Figure 4.23) are composed of many such chains.

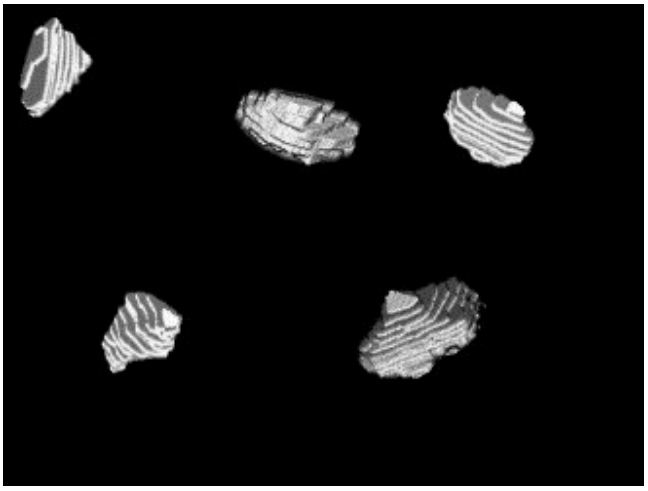


Figure 4.25: Three-dimensional morphologies of individual SiC particles extracted from the surface rendered three-dimensional image of the 8.1 PSR microstructure.

The information gained from the analysis of 3D microstructures of these composites has been utilized in generating *realistic* 2D simulations for these composites. The details of these procedures are provided in the next chapter.

# CHAPTER 5

# METHODOLOGY FOR SIMULATION OF 'REALISTIC' MICROSTRUCTURES

The central objective of this work is to develop a general, flexible, and practical methodology for simulations of 'realistic' microstructures at any length scale of interest that are suitable for implementation in the computational models and simulations for reliable predictions of materials behavior. A novel digital image processing based methodology for such simulations is presented in this Chapter via its applications to the microstructures of SiC particles in Al-alloy matrix composites and TiB whiskers in boron modified Ti-alloys. The methodology incorporates digital images of individual particles/features (that may have complex shapes/morphologies) from binary images of the corresponding real microstructures in the microstructure simulations. Further, the spatial arrangements (clustering, etc), morphological orientations, size-shape distributions, and relative amounts of the features in the simulated microstructures are statistically similar to those in the corresponding real microstructures, and therefore, the simulated microstructures are "realistic." The methodology is general but experimental quantitative microstructural data are required on the material of interest for simulations of corresponding realistic microstructures. In the present work, the detailed quantitative microstructural data reported in the previous Chapter are utilized for this purpose. The following sections of this Chapter provide the details of this technique. Furthermore, this methodology is used in correlating the process parameters of the materials to their microstructures and utilizing those relationships to create 'virtual' microstructures, which can be analyzed to predict the material behavior without physically manufacturing them.

The results of the application of this simulation-based methodology to the materials under study along with the creation of virtual microstructures are described in the next chapter.

## 5.1 Overview of the Simulation Methodology

The primary idea behind the simulation methodology developed in this research is the extraction of real particle/features morphologies from a given microstructure and then placing those particles/features in a simulation space, in such a manner so as to generate an image that is statistically similar to the corresponding real microstructure. Conceptually, the simulation procedure is as follows. Consider the high magnification high-resolution microstructural images such as the one in Figure 5.1b. Such microstructural images obviously contain real particle morphologies. Now, consider a thought experiment, where large number (~ thousand, or more, if needed) of second phase particles are simply "plucked" out from microstructural images (capturing of real particle morphologies is explained in the next section) and stored in a box, such that the set is representative of the size, shape, and morphology distribution of the entire population of the particles/features in the real microstructure. Use of such realistic particle/feature shapes/morphologies is one of the most important steps towards generating realistic microstructures. To the best of author's knowledge, such realistic complex particle/features shapes/morphologies have not been used in any of the earlier microstructure simulations reported in the literature. Another important aspect of the present technique is that it enables the incorporation of size-shape distribution of the particles/features of interest that is statistically representative of the corresponding population in the real microstructure. Next, the particle centroids are simulated (as per

some specified spatial arrangement and number density) in a digitized simulation space where the pixel size is the same as that in the microstructural images from which the particles/features are plucked out. Finally, the box containing the plucked out particle images is thoroughly "shaked"; one particle image is taken out at random; and placed at a randomly selected point in the simulation space. All pixels in the particle are then marked as binary dark pixels. This completes the placement of one particle/feature in the simulation space. The process is repeated by placing other particles/features till desired volume fraction and size/shape distribution of the particles/features is achieved. This results in a simulated microstructure having the same volume fraction, number density, and size-shape distribution of the particles/features of interest as those in the corresponding real microstructure. Nonetheless, at this stage, the spatial arrangements of the features/particles of interest are not necessarily the same as those in the corresponding real microstructure. To ensure a statistically similar spatial arrangement, it is essential to compute and match the statistical descriptors such as n-point probability functions, lineal path probability distributions, nearest neighbor distribution, radial distribution functions, etc., in the simulated microstructure and with corresponding experimentally measured statistical descriptors. If the match is not satisfactory, the simulation parameters are varied and the process is repeated (in the spirit of Monte Carlo techniques) till a satisfactory match between experimental and simulated statistical correlation functions is achieved, which then yields a simulated microstructure that is statistically similar to the corresponding real microstructure. Consequently, to simulate a realistic microstructure, the required experimental inputs are (i) the digital images of particles/features covering the entire range of size and shape distribution present in the real microstructure, and (ii)

the statistical correlation functions in the corresponding real microstructure. The technique can also be used for simulations of "virtual" microstructure using the same realistic particle images but any other desired volume fraction, size-shape distribution, and spatial arrangements of the features of interest. The following sub sections describe the important steps of the new microstructure simulation methodology.

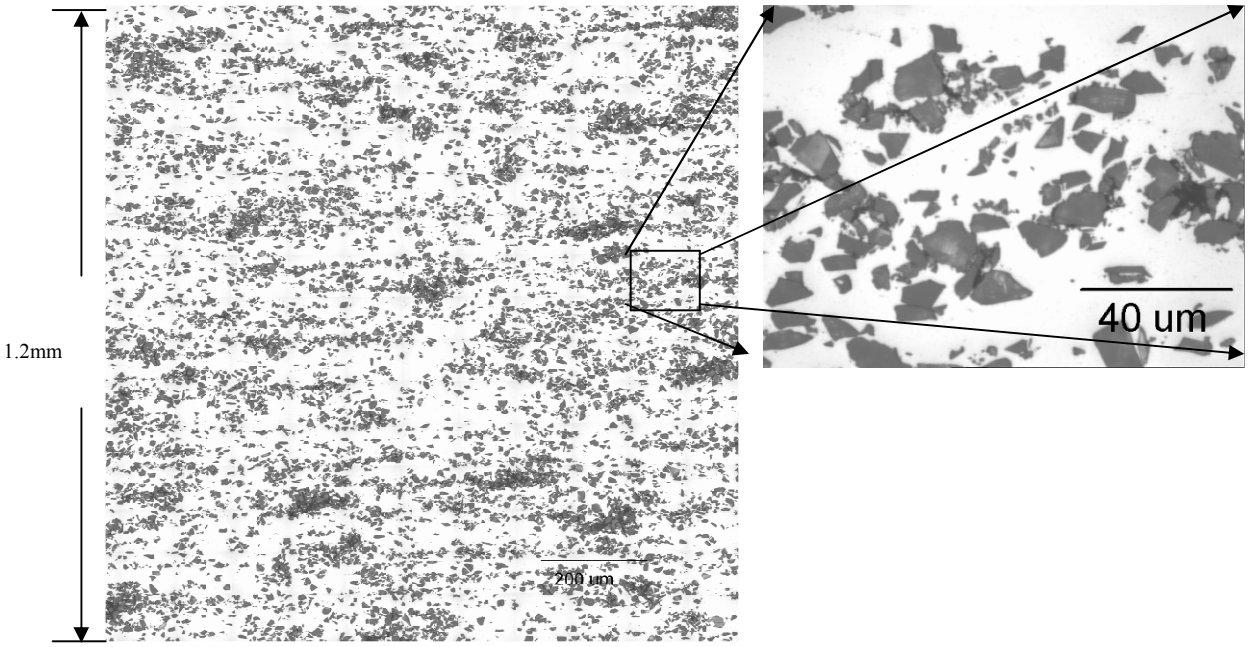



Figure 5.1: (a) Montage of Real DRA microstructure, (b) Single field of real microstructure for DRA specimen.

5.1.1. Capturing Real Particle Morphologies

The first step towards simulating "realistic" microstructures is capturing the real particle morphologies from a given microstructure. The set of (X, Y) coordinates of closely spaced points (pixels) on the boundary of the binary image of a particle/feature contains complete detailed information on the morphology and geometry of that particle/feature. Once such a set of boundary/contour points is available, the exact replica of that particle can be then reproduced at any desired location in the simulation space.

Recently, Ren, Yang, and Sun [106] have given an image analysis procedure to extract the boundary contours of the features in a binary digital image, which can yield the (X, Y) coordinates (pixel positions, to be more precise) of the closely spaced points (pixels) on the boundary of the feature. The motivation behind their research was to develop an algorithm to be used in image analysis for pattern recognition and it has not been previously applied to capture the morphologies in the microstructures. A C++ computer code based on their algorithm has been written in the present work (see Appendix A). The computer code uses digital binary images of the microstructure. The code identifies pixels on the boundaries of a particle and creates "inner" and "outer" boundary contours. The boundary contour is classified as outer contour if it encloses the particle, and it is considered as inner contour if it encloses matrix and is surrounded by the particle. Extraction of the coordinates/positions of boundary pixels is a three-step process. First, a starting point is found on a boundary, and then the contour is followed pixel by pixel, and finally, the termination of the contour is identified. In this way, the code generates the set of coordinates/locations of the pixels that form the boundary of a particle. Using this computer code, the boundary contours of SiC particles and TiB whiskers were extracted from binary microstructural images of 2.0 PSR DRA composite and extruded boron modified Ti-alloys, respectively, to represent the size, shape, and morphology distribution of the SiC particles and TiB whiskers population observed metallographically. These data sets contain the positions of the boundary pixels of each particle/whisker. For example, Figure 5.2 show the microstructure of boron modified Ti-alloy and Figure 5.3 depicts the extracted TiB whiskers "plucked" from the real microstructure and placed in a regular array, which is a small part of our "box" of particle images. The centroid pixel position of

each particle/whisker is then computed from the positions of the boundary pixels. Using simple coordinate translation (change of origin), the (X, Y) positions of the pixels on the boundary contour are changed so that each particle centroid is at (0, 0) position. The data set corresponding to each particle is then stored in the computer memory. Next, each of N particles is assigned a distinctive number in the range of 1 to N to identify that particle; these identification numbers are assigned in a random manner, and they have no correlation to the size or shape of the particles. Once the data set representing the pixels of the particles and their ID numbers are stored in the memory, any particle can be placed at the desired location in the simulation space.

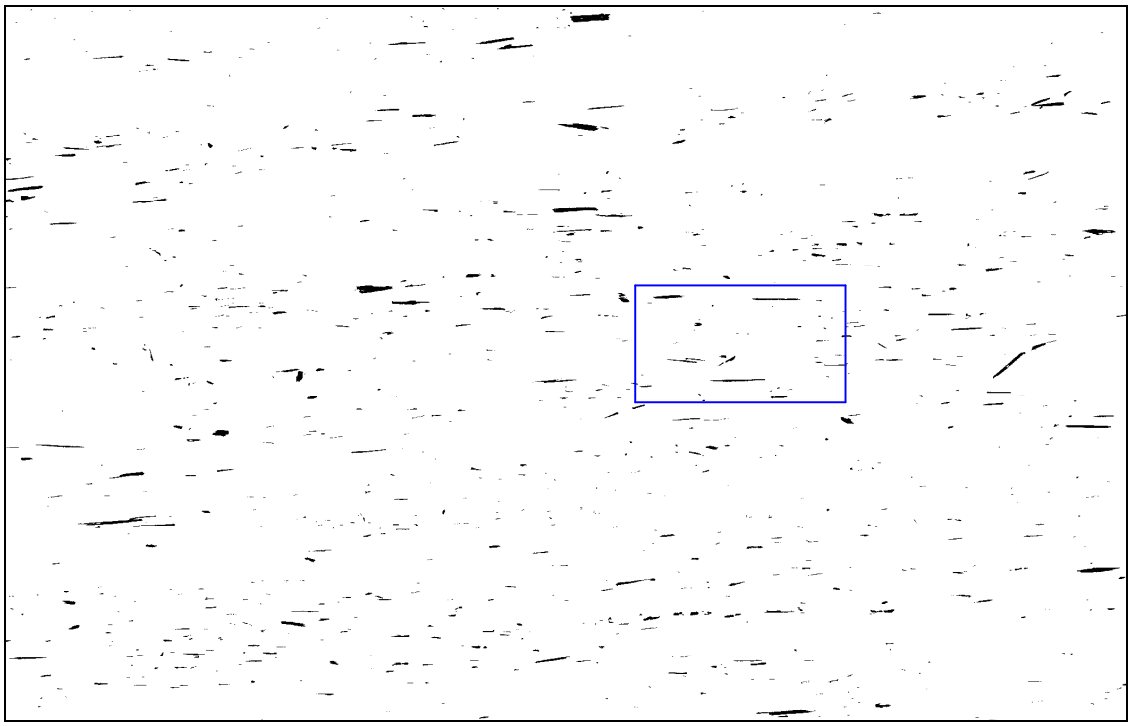


Figure 5.2a: Microstructure of boron modified Ti-alloy showing TiB whiskers.

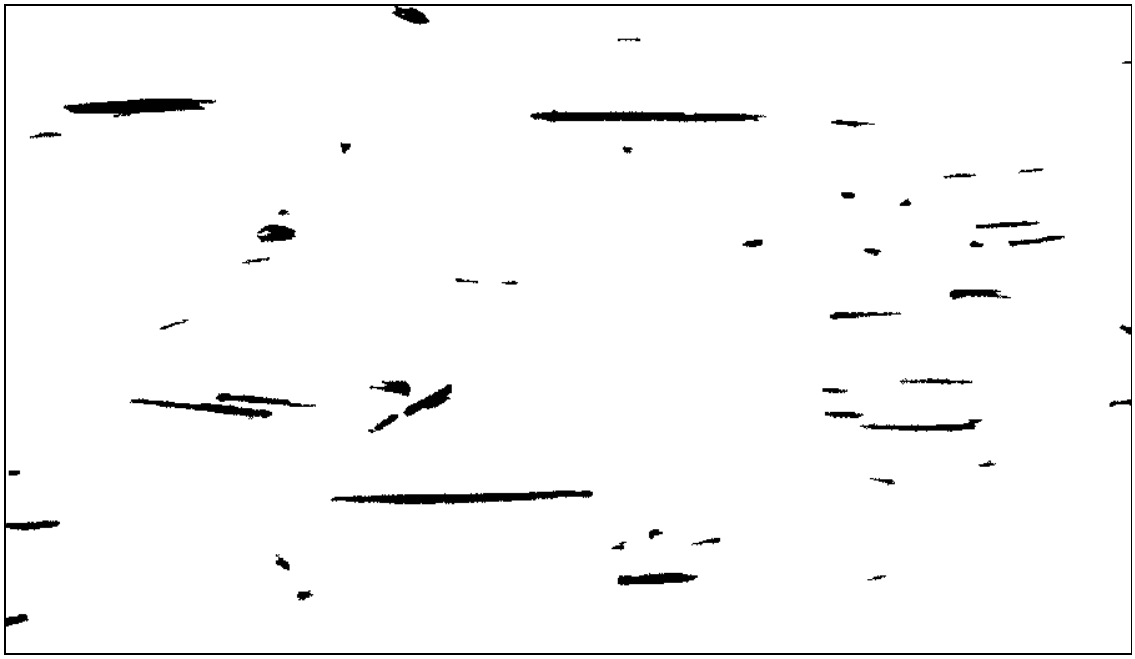


Figure 5.2b: High resolution image of the boxed region in (a), showing the morphology of TiB whiskers

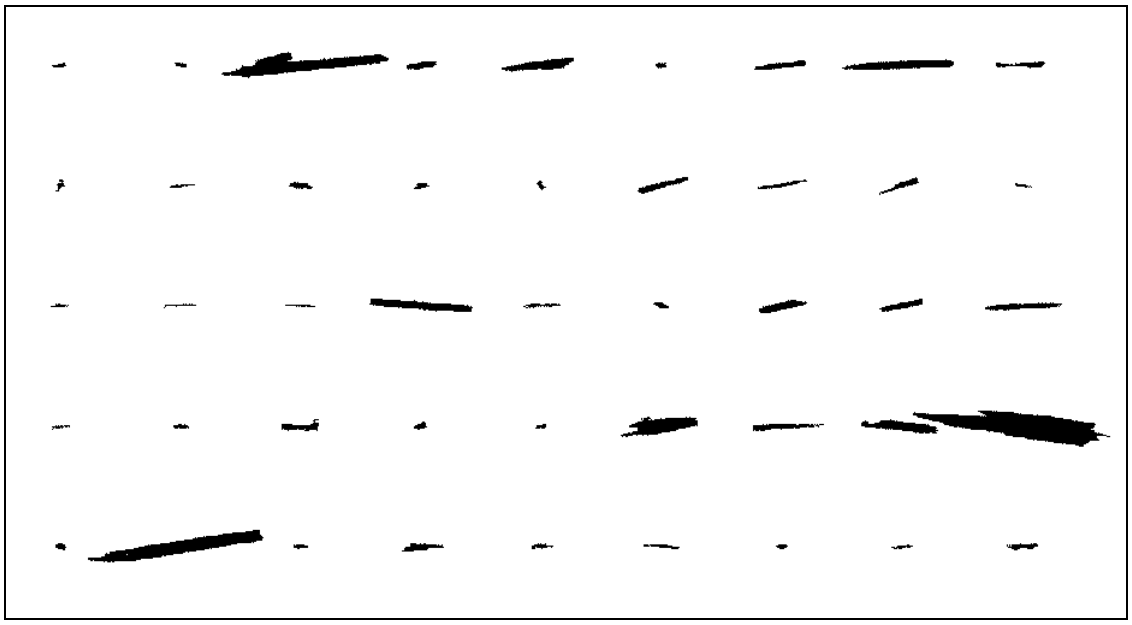


Figure 5.3: TiB whiskers extracted from real microstructure shown in Figure 5.2

### 5.1.2 Simulation of Locations of Particle Rich and Particle Poor Regions in the Simulation Space

As can be seen in Figure 5.1a, spatial clustering is an important microstructural feature in materials containing second phase particles. Furthermore, the clusters may themselves have a size/shape distribution rather than a single value of size and aspect ratio for all the clusters. Therefore it is essential to simulate the clustered regions that mimic the type of clusters present in the real microstructures. In the present material, the spatial clustering of SiC particles is primarily due to large values of the particle size ratio (PSR), whereas the directionality observed in the majority of the SiC particle clusters is primarily due to the extrusion process. Due to the extrusion process, majority of the particle clusters are elongated to large extent (i.e. high aspect ratio). Nonetheless, as the deformation due to the extrusion process is not necessarily uniform at all locations, some SiC particle clusters have low aspect ratio and they are relatively less elongated. Therefore, two types of cluster regions (representing SiC particle rich regions) are first simulated, namely, the high aspect ratio and low aspect ratio regions. The number densities of the two types of regions, their size, and aspect ratios are important simulation parameters. These regions are simulated using the well-known RSA algorithm [83]. The clusters are not permitted to overlap and their centers are at uniform random locations in the simulation space.

### 5.1.3. Simulation of Locations of Particle Centroids and placement of particles in the Simulation Space

Depending on the desired spatial arrangement of particle centroids, one can either simulate a uniform random spatial dispersion of points (representing particle centroids),

or create regions (which may represent particle clusters) having different point intensities (representing different number densities of the particle centroids) to simulate spatial clustering in the simulated microstructure. In the case of simulating microstructures having clustered regions of second phase particles, a parameter called *clustering intensity* is used to measure the extent of clustering. This parameter is measured as $([N_A]_{cluster}/[N_A]_{global})$, where $[N_A]_{cluster}$ is the average number density of the reinforcement particles inside the clustered regions and $[N_A]_{global}$ is global average number density of the reinforcement particles in the simulated microstructure. Once the type of realization required is specified, each particle centroid simulation is followed by the placement of the second phase particles on that centroid. The placement of a particle at a location in the simulation space simply involves translation of the particle centroid from (0, 0) to the pixel coordinates of the new location in the simulation space. Before placing a particle at the simulated centroid, it is checked for the overlap it may cause with the existing particles on the simulation space. The computer code has been designed with the flexibility of placing the particles without allowing any overlap (as in RSA algorithm), a controlled overlap similar to the cherry pit model [50] or allow them to overlap freely (as in Boolean schemes [85] and Gibb's process [87]), as needed for different types of microstructures. This process of simulating a centroid and placement of particles is iterated until desired overall volume fraction and size/shape distribution of particles is achieved.

Another important aspect while placing the particles in the simulation space is the change in orientation. Note that in the case of simulation of DRA composites, the particle orientations are kept the same as those in the corresponding real microstructure, as the

SiC particle shapes are equiaxed. However, the computer code can permit particle rotations, if needed. This feature has been used to simulate the microstructures of boron modified Ti-alloys having specified morphological anisotropy. The details of steps involved in rotation of particles are presented later in this Chapter.

### 5.1.4. Comparison of Statistical Correlation Functions of Simulated and Real Microstructures

For a chosen size and shape distribution of second phase particles and their volume fraction, the simulation parameters that can be changed to vary the microstructure are as follows.

(i) Number densities of elongated and equiaxed particle rich regions

(ii) Size, aspect ratios, and orientations of elongated and equiaxed particle rich regions

(iii) Clustering intensity, i.e., $([N_A]_{cluster}/[N_A]_{global})$

The microstructure can be altered by changing these parameters and hence two-point functions and the lineal path distribution functions of the simulated microstructure can be changed. These parameters are varied until a match between statistical co-relation functions of real and simulated microstructure achieved. In the present technique, it is necessary to begin with a set of "guess" values for the above simulation parameters. A microstructure is then simulated with that combination of the simulation parameters, and its two-point correlation function $P_{11}$ (r, θ) and lineal path distribution function $L_{11}$ (r, θ) is computed for different values of r ranging from 1 μm to 500 μm, and for different line orientations θ. The simulated correlation functions, $[P_{11}(r,\theta)]_{sim}$ and $[L_{11}(r,\theta)]_{sim}$ are then compared with the corresponding experimentally measured functions $[P_{11}(r, \theta)]_{exp}$ and

$[L_{11}(r, \theta)]_{exp}$. At any given value of r and θ, the absolute fractional error $E_P(r, \theta)$ and $E_L(r, \theta)$ can be computed as follows.

$$E_P(r, \theta) = \left| [P_{11}(r, \theta)]_{sim} - [P_{11}(r, \theta)]_{exp} \right| / [P_{11}(r, \theta)]_{exp} \quad (5.1)$$

$$E_L(r, \theta) = \left| [P_{11}(r, \theta)]_{sim} - [P_{11}(r, \theta)]_{exp} \right| / [P_{11}(r, \theta)]_{exp} \quad (5.2)$$

Let $< E_P(\theta) >$ and $< E_L(\theta) >$ be the average value of $E_P(r, \theta)$ and $E_L(r, \theta)$, respectively, averaged over all values of r for given direction (i.e., θ). It is well known that microstructures are not deterministic; rather they are of stochastic nature. As a consequence, microstructures of two composites having exactly the same chemistry, exactly the same raw materials, and exactly the processing conditions are not exactly the same (although they may be statistically similar). Further, there is always some random sampling error associated with estimation of any microstructural parameter. These effects can result in the variations of few percentages in any estimated microstructural attribute (including correlation functions). As the experimental data and microstructure has this level of variability it serves no purpose to try to match the correlation functions of the simulated microstructure to the experimental data to a level better than that. Similarly, it must be kept in mind that the simulations also have a stochastic component to them (just like real microstructures). The stochastic aspect of the simulations is due to (a) assignment of particle centroids within the region of interest (say particle clusters, etc) using a random number generator, and (b) placement of particle of given size and shape at that location, again using a random number. As a result, two realizations of the simulated microstructure having exactly the same simulation parameters are not exactly the same. Due to these stochastic components, the microstructure of real microstructure and the corresponding simulation can never be identical. It must be pointed out that in the

present work, the simulated microstructure is varied till it matches the corresponding correlation function of the real microstructure within certain limits. This is trial and error process intrinsic to all Monte Carlo class of simulations. Therefore, it is absolutely essential to work with some range within which the correlation function of the simulated microstructure should match those of the corresponding real microstructure. These limits must be chosen keeping in mind random sampling errors involved in the experimental measurements of correlation functions (typically on the order of 3 to 5%), stochastic variations in the real microstructures from one region to another, and the typical stochastic noise in the simulations (on the order of 3 to 5%). Therefore, the correlation functions of the real and simulated microstructures can be matched only up to these levels. Therefore, in the present work, a simulated microstructure is considered to be representative of the corresponding real microstructure, if and only if, (1) $< E_P (\theta) >$ and $< E_L (\theta) >$ are less than or equal to 0.05 for each direction $\theta$ of interest, and (2) For any give value of r and $\theta$, $E_P (r, \theta)$ and $E_L (r, \theta)$ are less than or equal to 0.07. Thus, the process involves numerous iterations of simulated microstructures with different combinations of the simulation parameters till the above two conditions are satisfied. This methodology has been applied to simulate SiC particles in the DRA composites and TiB whiskers in the boron modified Ti-alloys. The details of these simulations are provided in the next chapter.

### 5.2. Simulations of Microstructures Having Specified Morphological Anisotropy

The above technique is used to simulate the microstructures where the particle shapes are equiaxed such as those of the SiC particles in the DRA composites. On the

hand, in numerous microstructures the constituent phases have preferred morphological orientations leading to partially or completely anisotropic microstructures. This section details the additional steps involved in simulation of 'realistic' microstructures having specified morphological anisotropy.

Consider simulation of a microstructure with a variable extent of morphological anisotropy using particle/whisker images extracted from the image of the microstructure in Figure 5.2. Each whisker image needs to be rotated by a specified amount before placing it in the simulation grid space. For this purpose, each pixel belonging to that whisker has to be rotated by the desired angle. Elementary geometric considerations yield following expression for new coordinates $(X_n, Y_n)$ of a *pixel* after rotation by an angle $\alpha$ that has original coordinates (X, Y).

$$X_n = R\cos[\theta + \alpha] \quad \text{and} \quad Y_n = R\sin[\theta + \alpha] \tag{5.3}$$

where, $R = [X^2 + Y^2]^{1/2}$ and $\theta = \tan^{-1}[Y/X]$

In a digital image, the pixel coordinates can only take integer values. Therefore the new coordinates $X_n$ and $Y_n$ have to be rounded off to nearest integers. This rounding off of the new coordinates to integer values can generate some empty (black) pixels inside the rotated particle/whisker (white). To take care of this problem, after the coordinate transformation the rotated particle/whisker is scanned and each black pixel that is completely surrounded by white pixels is converted to a white pixel. Once the scanning is complete the correct pixilated image of the rotated particle/whisker is obtained. Note that the rotation of the particle/whisker does not change the shape or size of the original particle/whisker. The rotated particle/whisker is then translated to its new centroid location in the simulation grid space. This technique can be applied to generate

microstructure with different level of morphological anisotropies; details of which are given in the following sub sections. An example of simulating a completely *isotropic virtual* microstructure using the whisker images from the *aligned* microstructure shown in Figure 5.2 is discussed below.

### 5.2.1 Simulations of Isotropic Microstructures

A completely *isotropic virtual* microstructure has been simulated using the whisker images from the *aligned* microstructure. As before, the first step is extraction sufficiently large (~ 1000 or so) number of whisker images from Figure 5.2. In an *isotropic* microstructure, the whisker orientations are uniform random. For such simulation, a random integer from 0 to 360 is chosen using a random number generator, and a given whisker is rotated by that many degrees. A new random number is selected for the second whisker image, and the process is repeated till all whiskers are rotated. The rotated whiskers are then placed at uniform random locations in the simulation space using the process in the previous sections. In the present simulations, whisker overlaps are not permitted, although it is feasible to permit specified degree of overlaps. Figure 5.4 depicts such a uniform random *isotropic* microstructure simulated using the realistic whiskers shapes/morphologies extracted from real aligned microstructure in Figure 5.2.

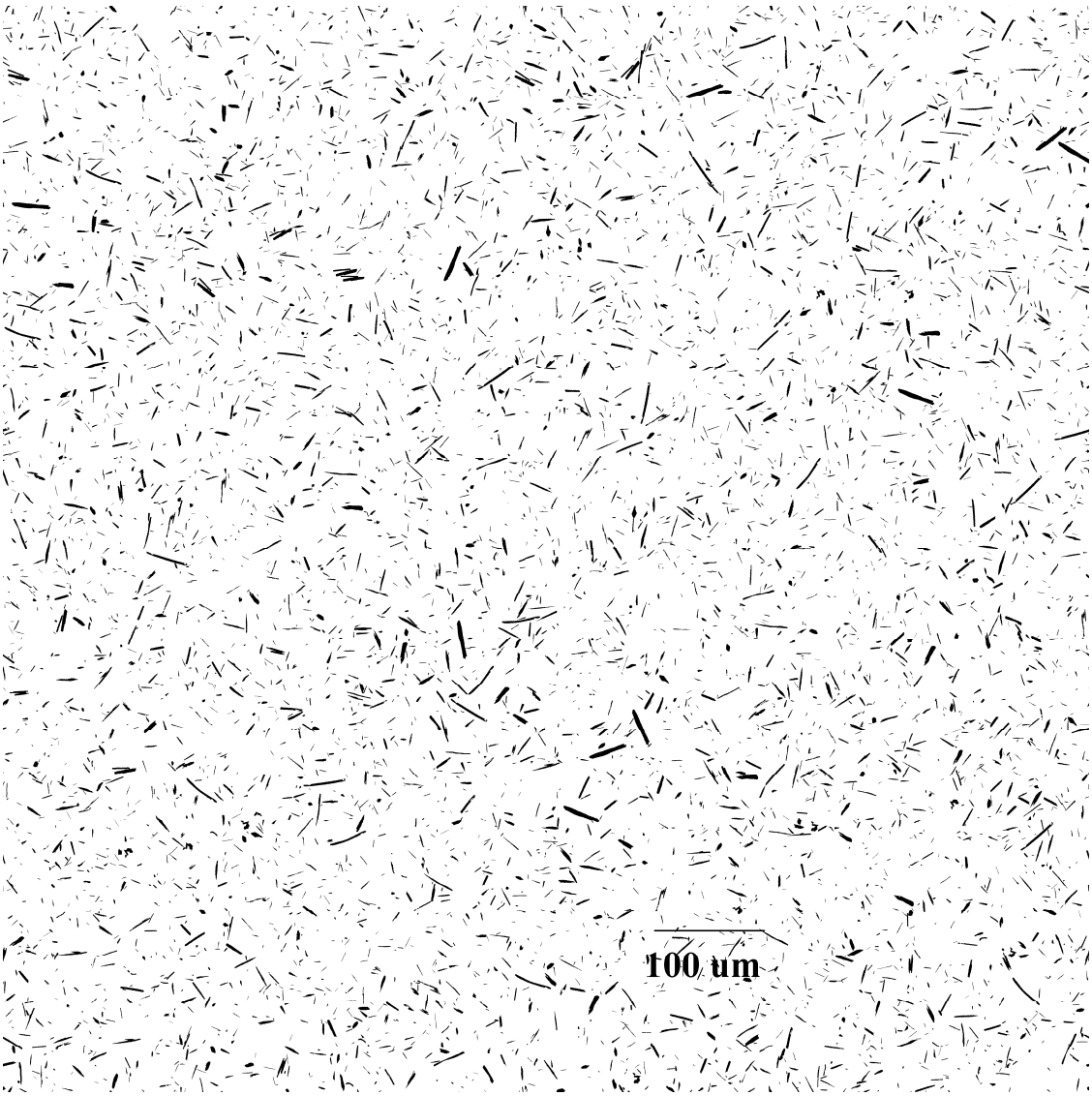



Figure 5.4a: Random isotropic microstructure simulated using the whiskers from the extruded boron modified Ti-6Al-4V alloy (Figure 5.2a).

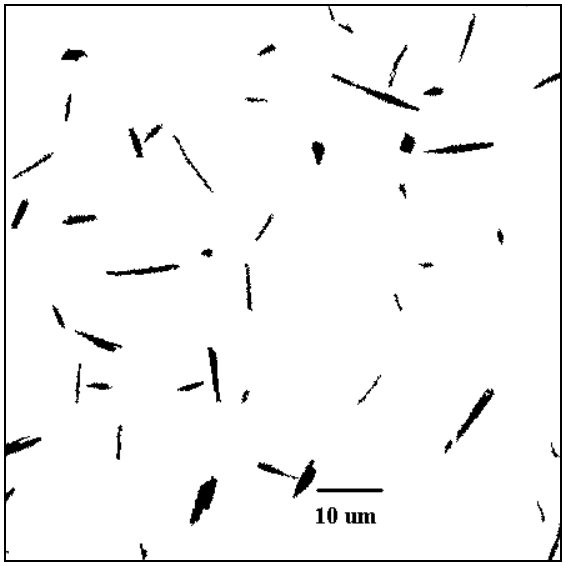



Figure 5.4b: High resolution image of simulated microstructure shown in Figure 5.4a showing randomly oriented TiB whiskers.

5.2.1 Simulations of Partially Anisotropic Microstructures

Using the whisker images from the uniform random isotropic microstructure in Figure 5.4a, a *partially* anisotropic microstructure have been simulated, which represents change in the microstructure due to deformation processing (say extrusion) of the alloy. For this purpose, each whisker image in Figure 5.4a is rotated by a certain angle. The original image of the whisker is then erased from the simulation grid and is replaced by the same whisker after rotating it by the given angle. The centroid of the rotated whisker is kept same as that of the original whisker image. Clearly, the amount of rotation given to each whisker depends on the physical process to be simulated and the corresponding microstructure model. *As an example*, consider a model where the revised angle of rotation for each individual whisker is given by the following equation.

$$\text{revised angle of rotation} = \beta + T_1 + T_2 + T_3 \qquad 5.1$$

where,

$$T_1 = \frac{(a - A)}{A} * C_1$$

$$T_2 = \frac{(s - S)}{S} * C_2$$

$$T_3 = \frac{(\theta - \Theta)}{\Theta} * C_3$$

In this equation, β represents the average angle of rotation, which is equal to the amount of rotation a whisker having average size, average aspect ratio, and average orientation will experience. A, S and Θ represent the average values of size (area), shape (aspect ratio) and orientation angle of the whiskers respectively and a, s and θ represent the size, shape and orientation angle of the particular whisker under consideration respectively. The constants $C_i$ are varied in an iterated manner by computing the orientation distribution in the simulation and comparing it with the target morphological orientation distribution needed. The final values for the constants $C_i$ are used such that the orientation distribution of the particles in the simulated microstructure represents the target distribution. In the present model, whiskers with above average aspect ratio, above average size and above average orientation angle from the deformation axis are rotated more than the whiskers with the average values of these attributes. Note that the same basic technique can be applied for any other models for amount of individual whisker rotation. Figure 5.5 shows the orientation-frequency graph for the TiB whiskers after the rotation model is applied to the microstructure shown in Figure 5.4. Figures 5.4 and 5.5 show simulated microstructures of the TiB whiskers in boron modified Ti-6Al-4V alloy

having different degrees of anisotropy generated using the model described by equation 5.1. Figure 5.4 represents an isotropic microstructure and Figure 5.6 depicts rotations of particles toward the deformation axis (in this case horizontal direction) as the isotropic sample undergoes simulated deformation processing. Note that Figure 5.6 is an example of a partially anisotropic microstructure, where whiskers do have a tendency to align along the deformation axis, but all the whiskers are not aligned parallel to the deformation axis as depicted in Figure 5.5 showing the orientation-frequency graph generated for TiB whiskers in the microstructure. This technique is useful in simulating microstructures where different extrusion ratios are used to produce microstructures with different levels of anisotropies. Using this methodology, a relationship can be established between the extrusion parameters and the simulation parameters. The details of the simulation of "realistic" microstructures and the results of the correlations between the processing parameters and the simulation parameters are provided in the next Chapter.

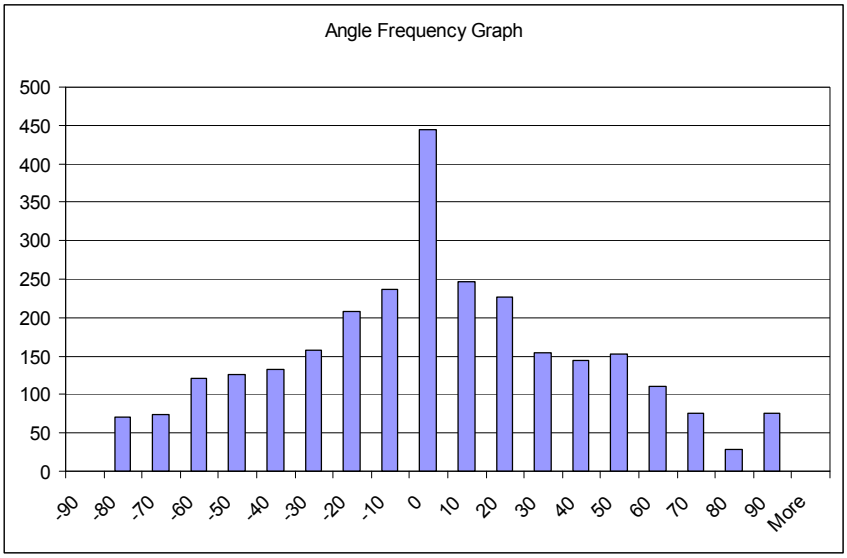



Figure 5.5: Orientation-frequency graph for the Ti-B whiskers in the microstructure shown in Figure 5.6.

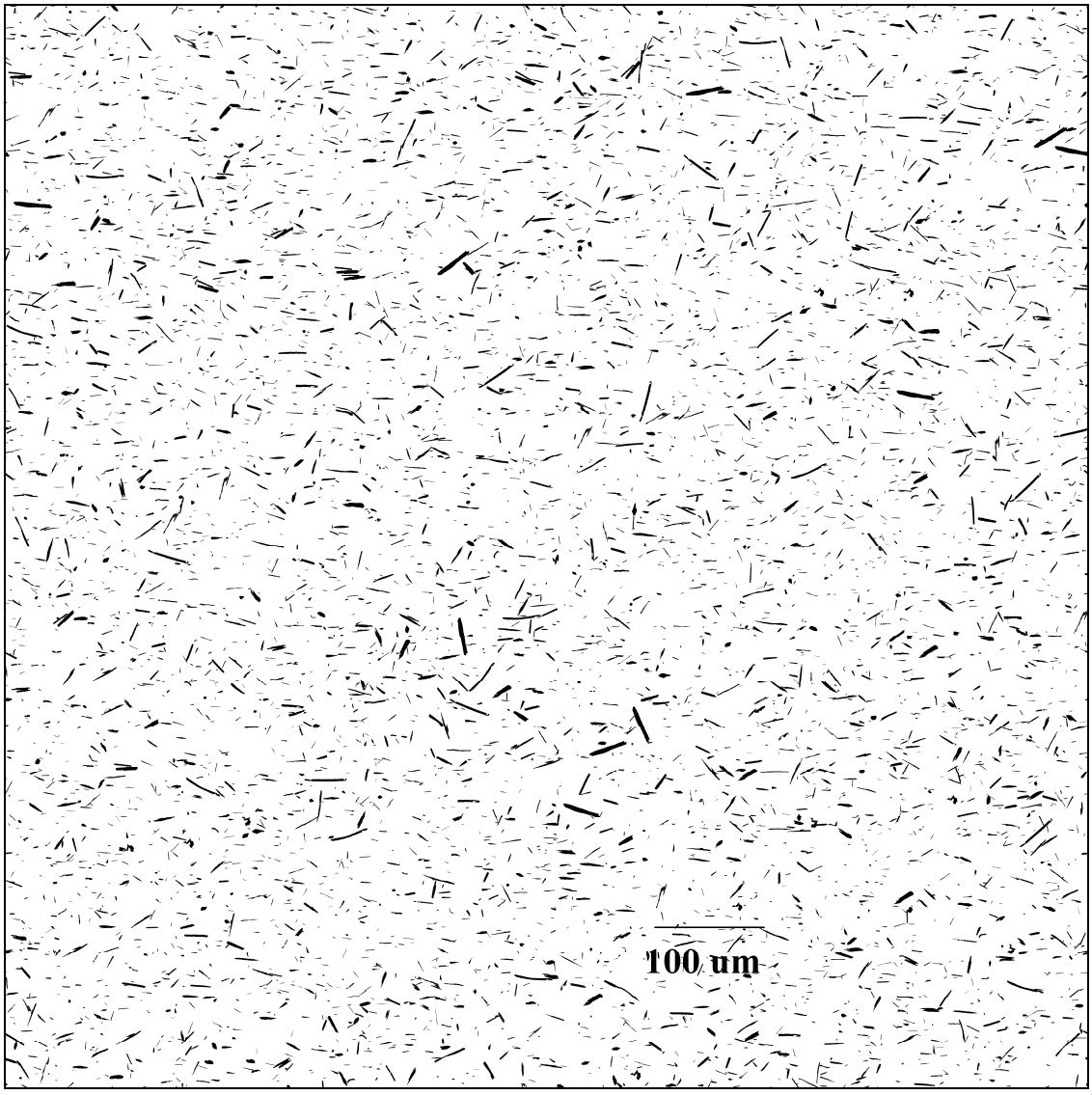



Figure 5.6: Simulated microstructure after simulating extrusion process on Figure 5.4a containing partially aligned whiskers.

# CHAPTER 6

# RESULTS OF COMPUTER SIMULATIONS OF MICROSTRUCTURES

The central objective of this work is to develop a methodology for simulations of 'realistic' microstructures. The general methodology for creating such simulations has been detailed in the previous Chapter. The results of these simulations are presented in this Chapter. Further, the simulations results have been utilized to develop quantitative correlations between the processing parameters and the simulation parameters, and these correlations have been utilized to generate an "atlas" of rational virtual microstructures that cover a wide range of processing conditions. Such virtual microstructures can be implemented as representative microstrucutral elements in the computational models/simulations of material properties/behavior to generate large simulated properties data sets that can provide critical input for microstructure-based materials design and development. The microstructure simulation results are presented in the next section, and that is followed by generation of rational virtual microstructures, and implications for microstructure-based materials design.

## 6.1 Simulations of "Realistic" Microstructures of DRA Composites

Three different samples of the DRA composites, each with different matrix to reinforcement particle size ratio (PSR), have been used for this work. As reported earlier, PSR is an important processing parameter that controls the spatial homogeneity of the reinforcement particle distribution in the composites manufactured via powder metallurgy route [1]. The Al-$SiC_P$ composites studied in this research have particle size

ratios (PSR) of 2.0, 3.1 and 8.1. Figure 6.1a to 6.3a show micrographs of the three DRA samples with PSR 2.0, 3.1 and 8.1, respectively. As it can be clearly noted, microstructure of sample with PSR 2.0 is relatively homogeneous, and as the PSR increases to 8.1, clusters of SiC particles can be seen in the extrusion direction. These microstructures have been utilized to study and quantify different levels of clustering in materials with same reinforcement particles and matrix alloy composition. The correlations of the extent of spatial clustering of SiC particles with the process parameter PSR have been studied via simulations of *realistic* microstructures.

The simulations for the microstructures of all the three specimens have been created using the technique described in the previous Chapter. Figures 6.1b to 6.3b show the simulated microstructures for the composites having PSR 2.0, 3.1 and 8.1, respectively. Comparison of Figures 6.1a to 6.3a depicting the real microstructures with the corresponding simulated microstructures in Figures 6.1b to 6.3b qualitatively reveals that the simulated microstructures appear to be statistically similar to the corresponding real microstructures with respect to the shapes/morphologies, spatial arrangement, size distribution, and volume fraction of the SiC particles. Detailed quantitative comparison of the real and simulated microstructures can be made using the statistical descriptors such as correlation functions. In the present work, two-point correlation functions and lineal path probability distribution have been used for this purpose[1]. Figures 6.4, 6.5 and 6.6 show the comparison of two-point probability functions measured in the real and simulated microstructures for PSR 2.0, 3.1 and 8.1 respectively. These functions have been measured in three different directions; (a) extrusion direction, (b) transverse

[1] Note that the present methodology is general, and therefore, one can also use other descriptors such as radial distributions, K-function, and nearest neighbor distributions for this purpose.

direction and (c) at an angle of 45 degrees between extrusion and transverse direction and they show a good match between real and simulated microstructures in all the measured directions: the average deviation between the data points of real and corresponding simulated microstructures is at the most 5%, and the deviation between any individual data point of real and corresponding simulated microstructure is at the most 7%. Figure 6.4d shows a plot of percentage error between real and simulated two point probability functions normalized by the maximum percentage of 6.52% for the specimen with PSR 2.0 in the extrusion direction. Figures 6.5d and 6.6d show the corresponding plots for specimens with PSR 3.1 and 8.1 respectively. Region to region spatial variations in the real microstructures of composites are usually of this order, and therefore, it is not necessary to match the two-point correlation functions of simulated and real microstructures to a better accuracy. The statistical similarity of the real and simulated microstructures can be further validated using the lineal path probability distributions because the lineal path probability distributions and two-point correlations functions are in general independent of one another (i.e., a lineal path probability distribution can not be computed from a given two-point correlation function and vice versa, in general). Figures 6.7, 6.8 and 6.9 compare the lineal path probability distributions of the real and simulated microstructures. The agreement between these data of the real and simulated microstructures validates the statistical similarity of these microstructures. Short range matching of these functions confirms the similarity of particle sizes and shapes and these descriptors at longer distances match the spatial distribution of particles in real and simulated microstructures.

The statistical similarity between the real and simulated microstructures essentially implies that the microstructural geometry of these microstructures can be conveniently represented in terms of the set of simulation parameters used for generation of these realistic simulated microstructures. This is because this input of the simulation parameters is sufficient to recreate/simulate a microstructure statistically similar to the corresponding real microstructure. In the present case, cluster intensity, cluster sizes and shapes, number density of clusters, extent of particle overlap, and size-shape distribution and volume fraction of the SiC particles are the simulation parameters. The values of the simulation parameters used to generate the realistic simulations of the DRA composites are given in the Table 6.1. The overlap between particles is observed in the real microstructures is due to the fact that the SiC particles tend to sinter during processing. The physical act of sintering of the reinforcement particles is represented by the use of overlap parameter in the simulation of these microstructures. Inspection of Table 6.1, showing the other simulation parameters, reveals that the three microstructures having different spatial clustering of SiC particles due to different PSR (a process parameter) values have been simulated by changing the value of just one simulation parameter, namely, clustering intensity; all other simulation parameters have the same values for the three simulated microstructures. Since all the processing parameters, except the PSR, were kept the same during the processing of the present DRA composites, the changes in the simulation parameter, clustering intensity, essentially represent the variations in the microstructure due to the changes in the process parameter, PSR. As expected, the increase in the level of clustering with increasing PSR is represented by relationship between clustering intensity and PSR shown in Figure 6.10, which depicts the increase in

clustering intensity with the increasing PSR. Now, once the relationship between clustering intensity and PSR has been established, it can be utilized, in combination of other simulation parameters, to generate a series of rational virtual microstructures with varying values of PSR.

Note that, the other simulation parameters can also be directly linked to the actual processing of the DRA composites. Variations in the extrusion parameters can be mimicked by the change in the simulation parameters of cluster shape/size and number density of clusters; increasing the extrusion ratio will increase the aspect ratio of the clusters while decreasing the overall number density of the clusters and vice versa. Further, the change in PSR can also be translated to the simulation model by the change in the average size of the SiC particles used to generate the simulated microstructures while keeping the matrix particle size fixed, as opposed to changing the matrix particle size with a fixed set of SiC particles, which is the case in the real DRA specimens studied in this research.

It can be concluded that changes in each of the simulation parameters mimics the microstructural variations due to some specific process parameters. Therefore, correlations between the simulation parameters and the corresponding process parameters can be utilized to generate an "atlas" of rational virtual microstructures that cover a wide range of processing conditions. The details of such virtual microstructures are presented in the following section.

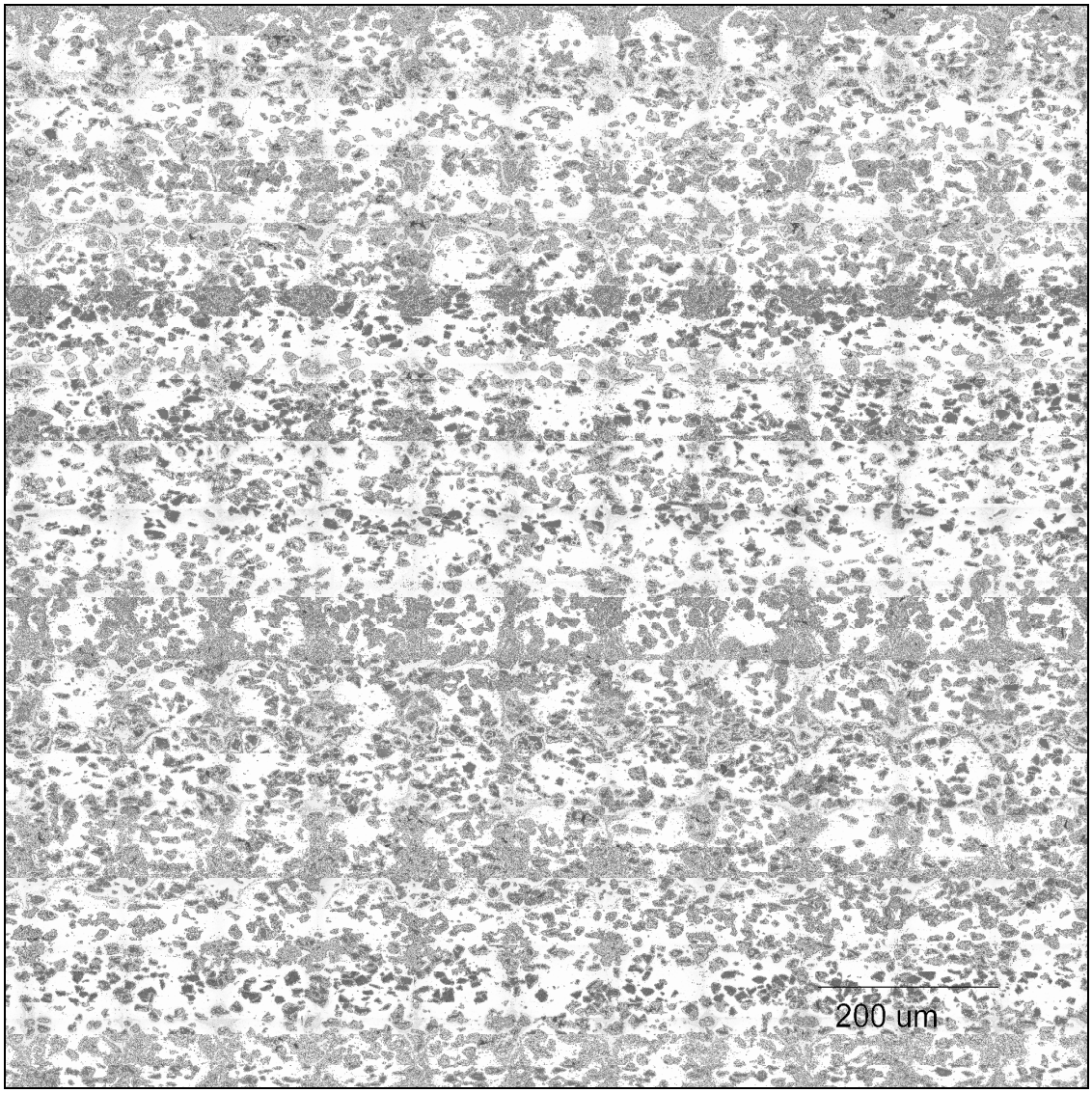



Figure 6.1: (a) Micrograph showing real microstructure for DRA composite with PSR 2.1

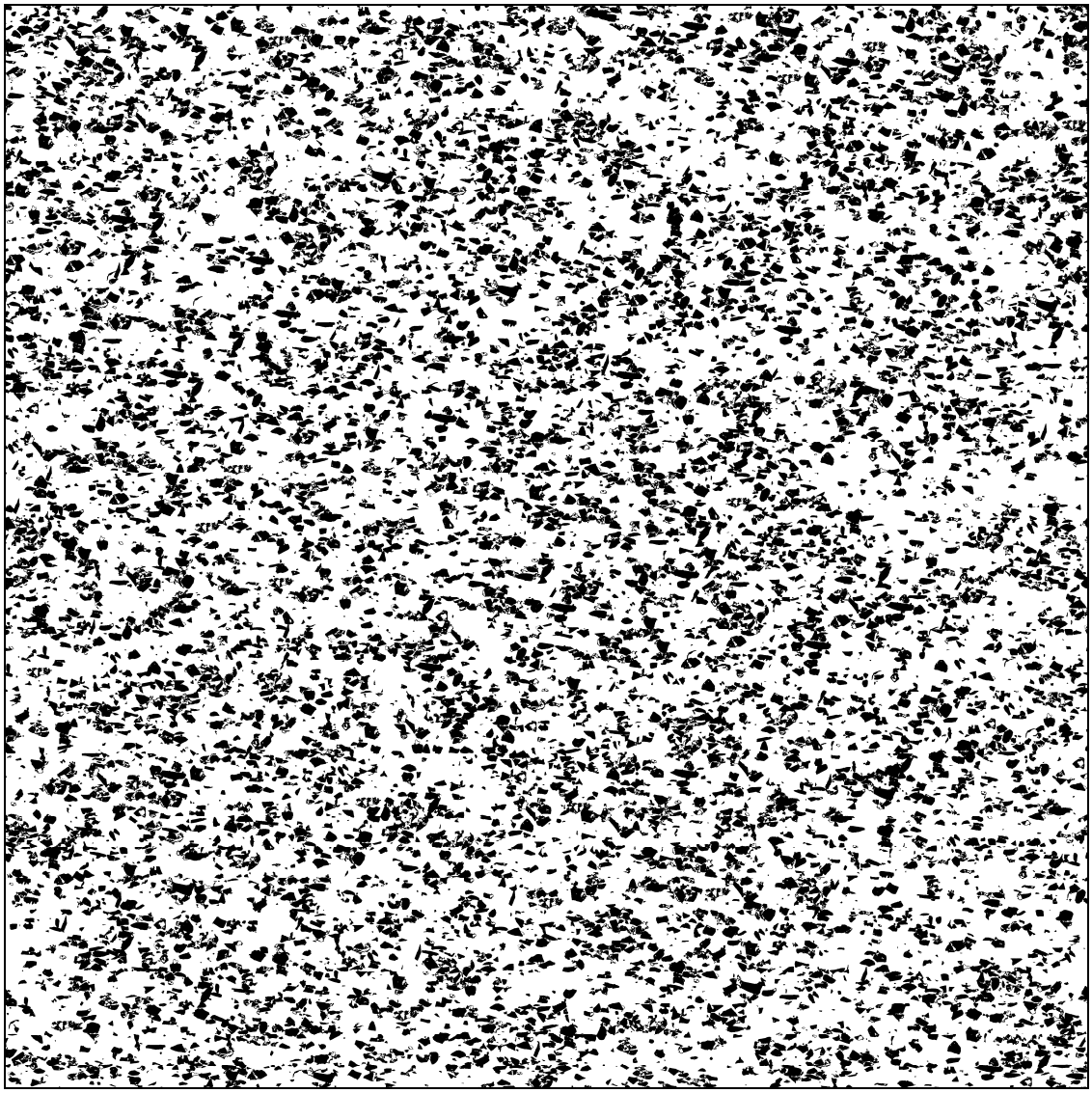


Figure 6.1: (b) Micrograph showing simulated microstructure for DRA composite with PSR 2.1.

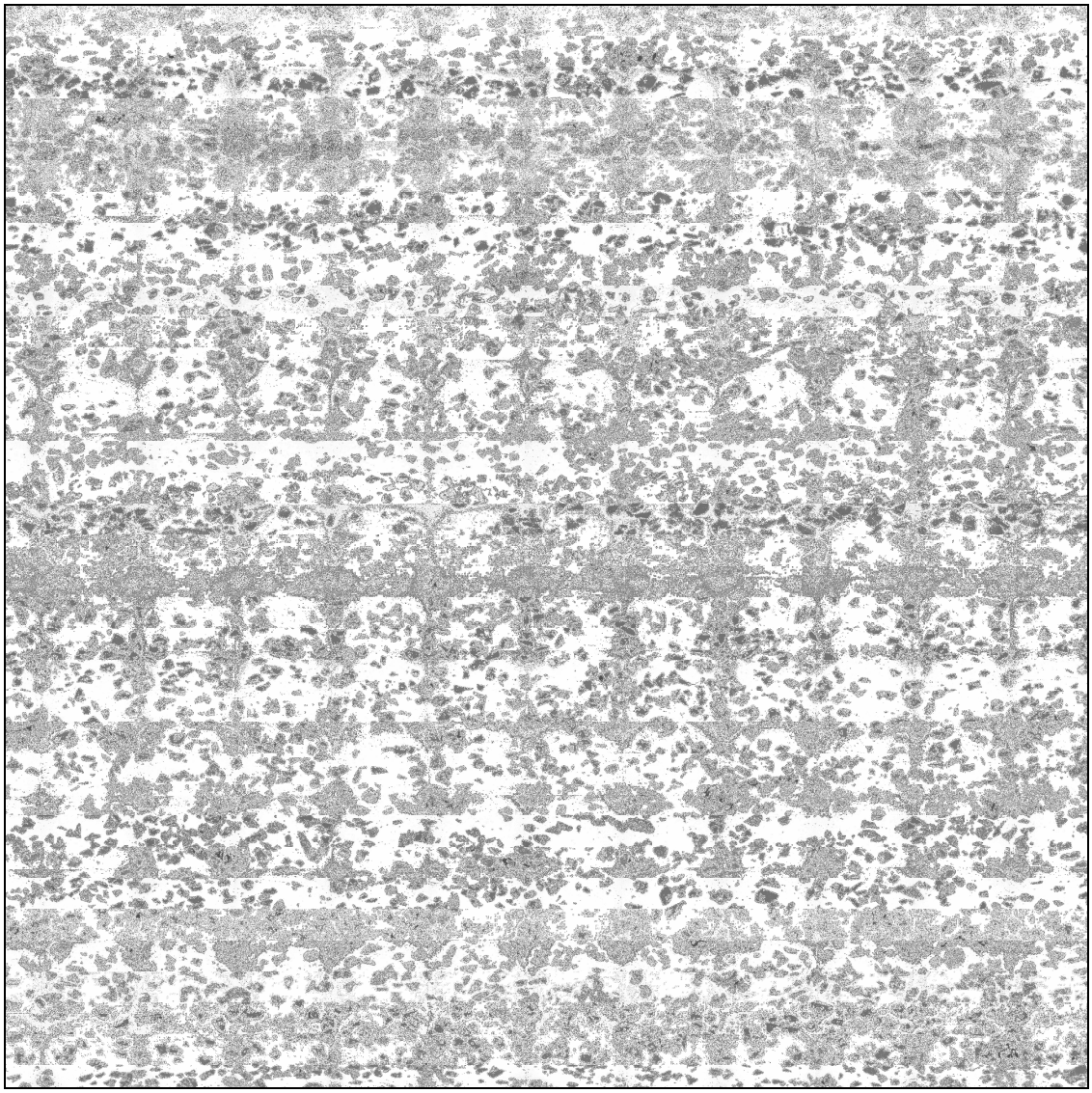


Figure 6.2: (a) Micrograph showing real microstructure for DRA composite with PSR 3.1.

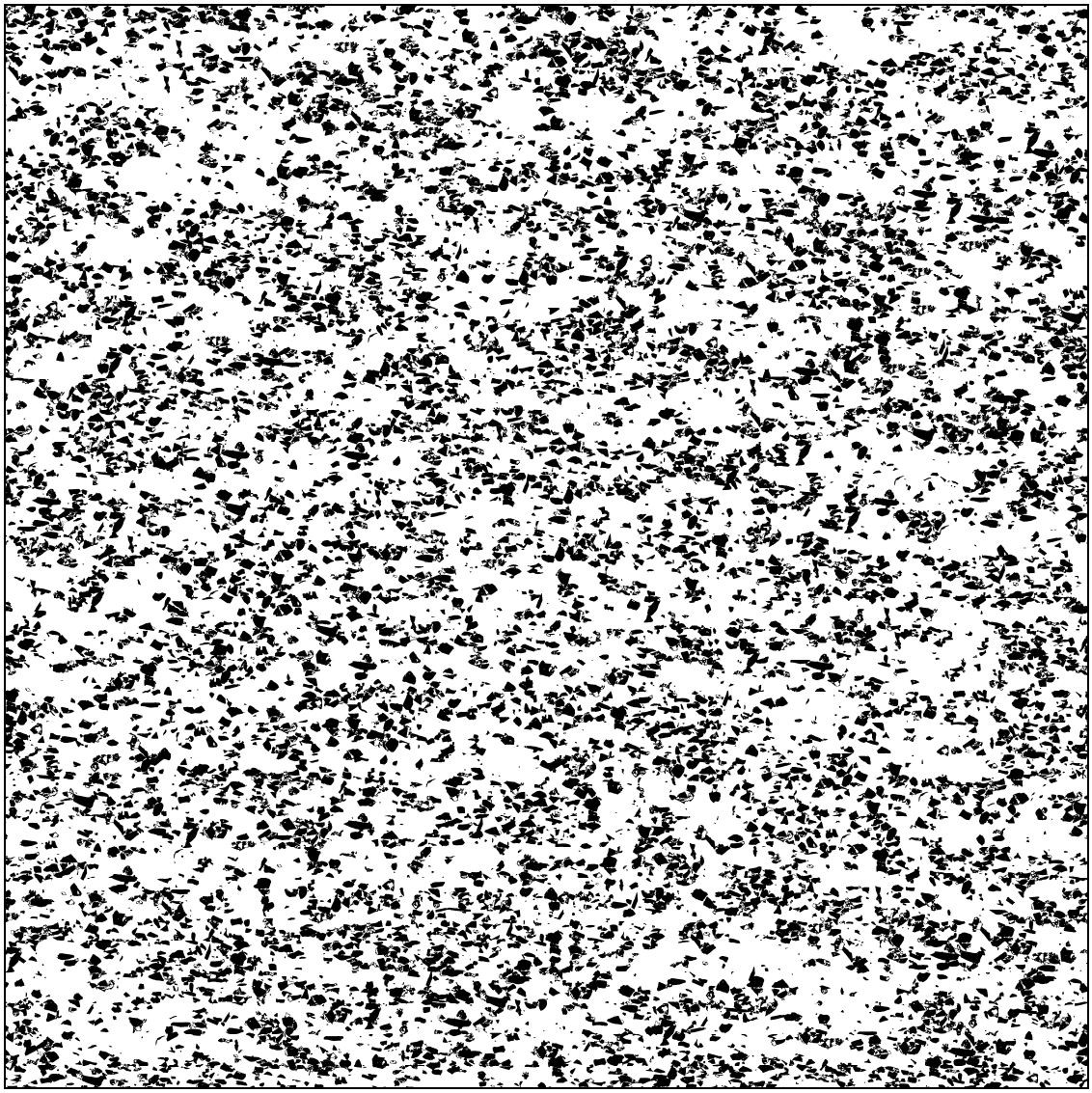


Figure 6.2: (b) Micrograph showing simulated microstructure for DRA composite with PSR 3.1.

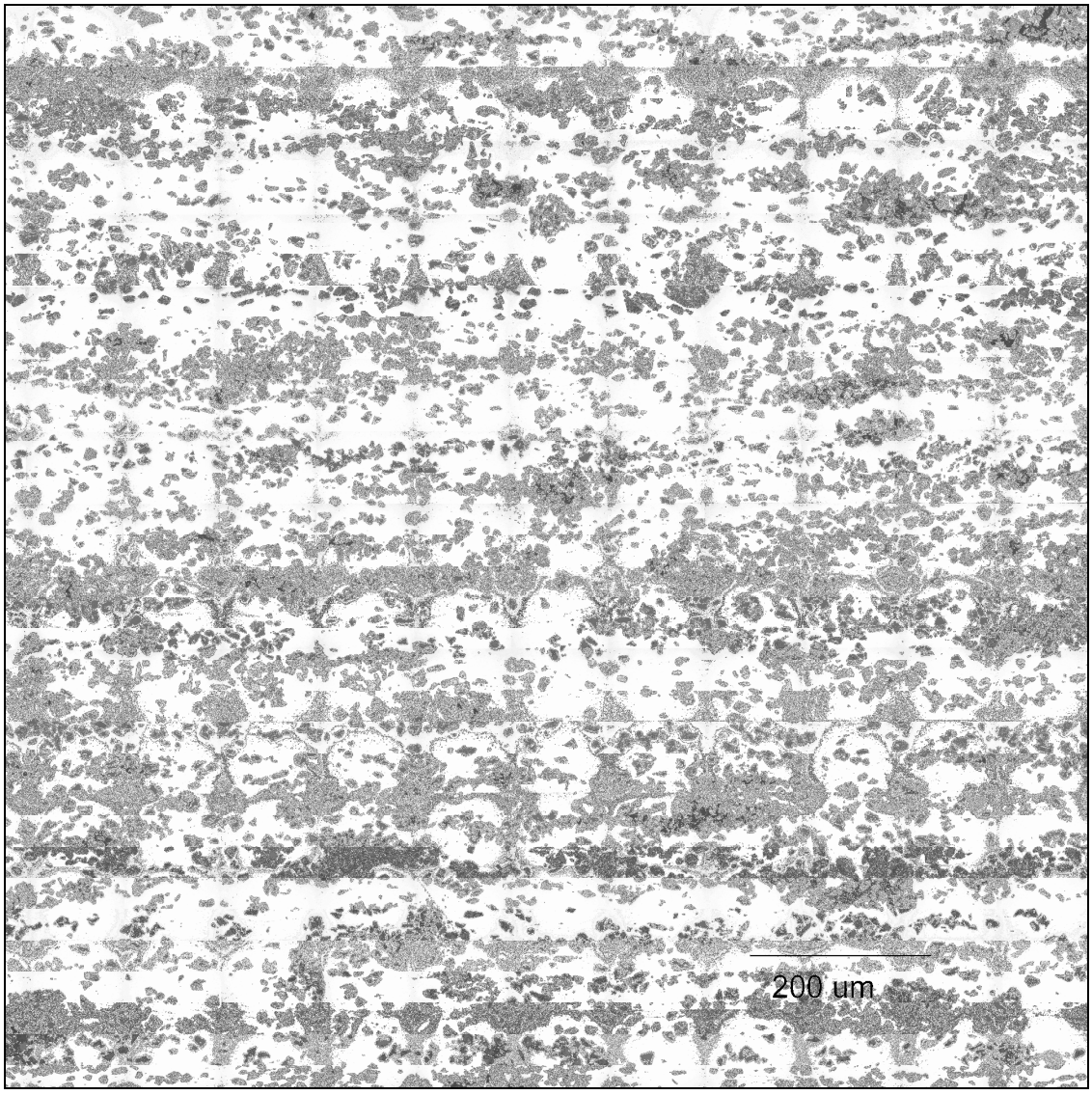



Figure 6.3: (a) Micrograph showing real microstructure for DRA composite with PSR 8.1.

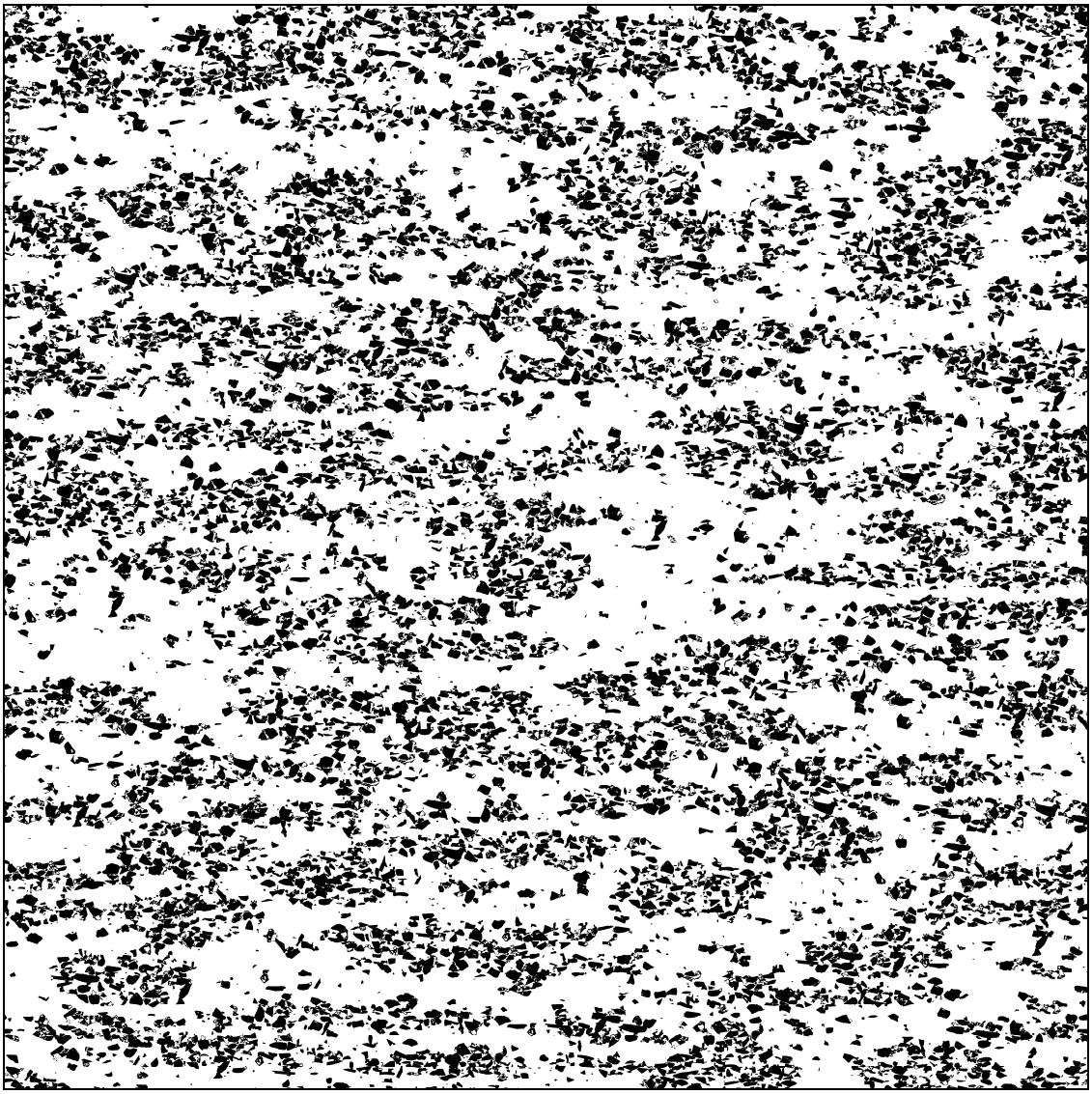


Figure 6.3: (b) Micrograph showing simulated microstructure for DRA composite with PSR 8.1

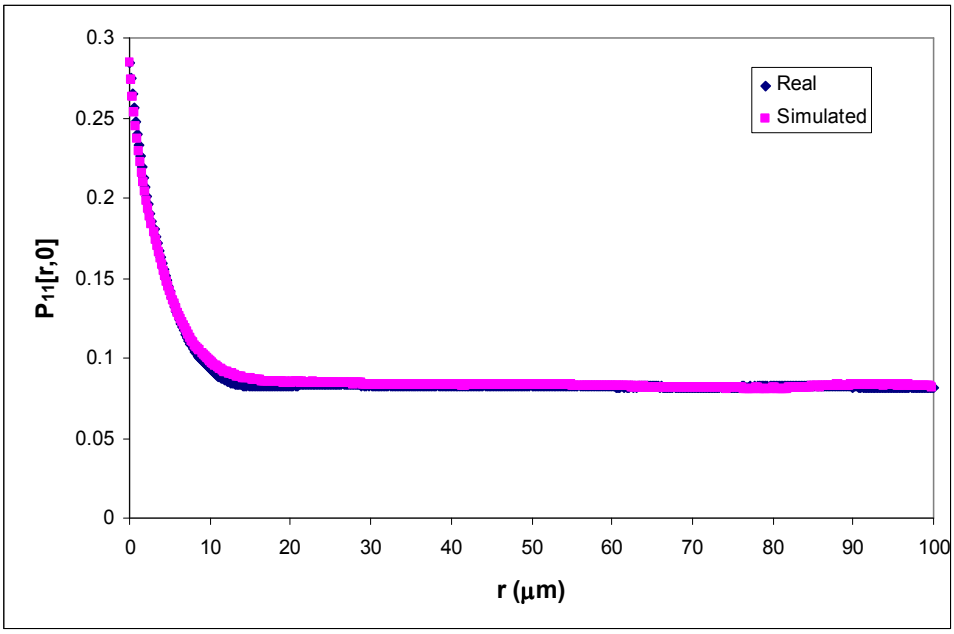



Figure 6.4a: Comparison of two-point distribution functions measured in the extrusion direction for real and simulated microstructures of DRA composites with PSR 2.0.

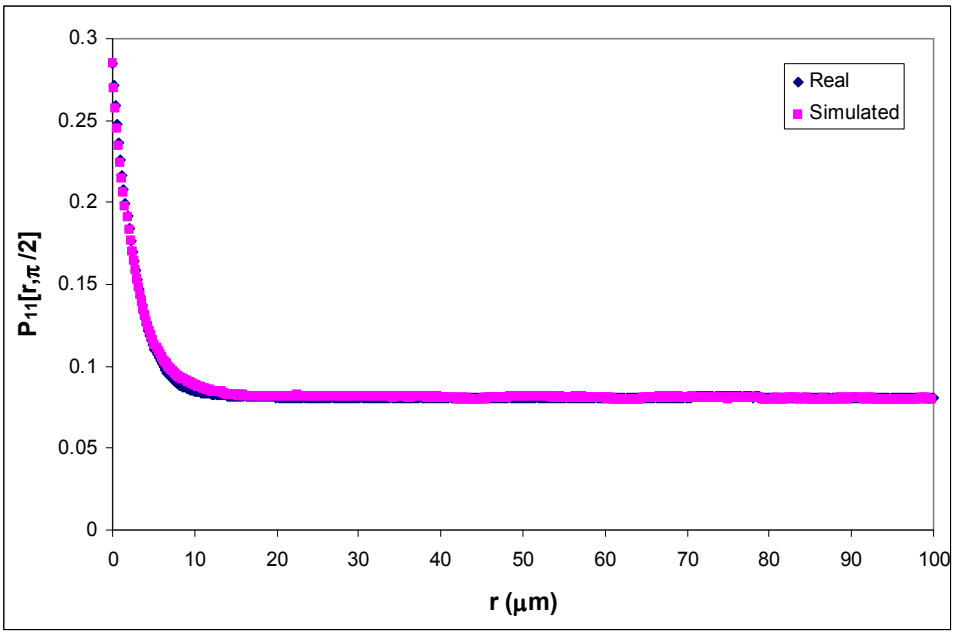



Figure 6.4b: Comparison of two-point distribution functions measured in the transverse direction for real and simulated microstructures of DRA composites with PSR 2.0.

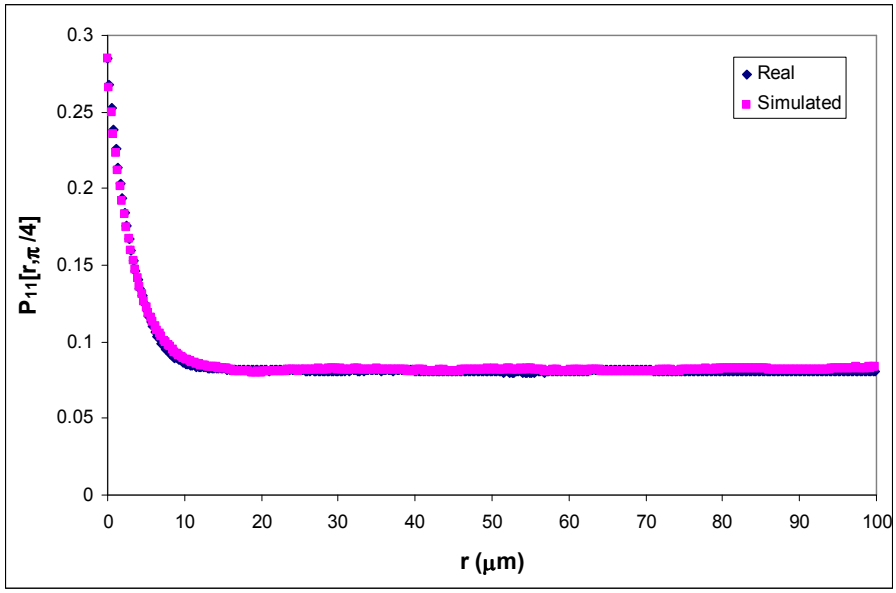



Figure 6.4c: Comparison of two-point distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of DRA composites with PSR 2.0.

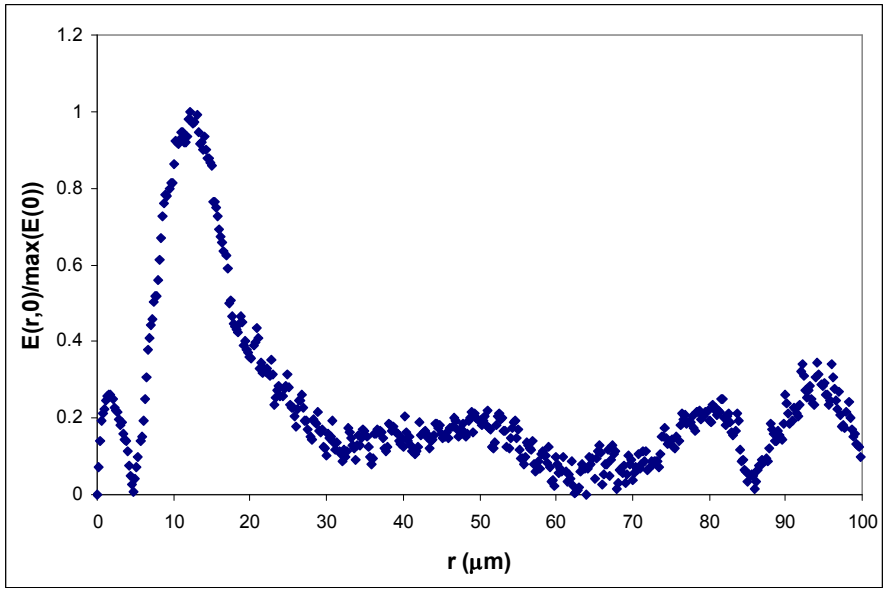



Figure 6.4d: Plot of percentage error between real and simulated two point probability functions normalized by the maximum percentage of 6.52% for PSR 2.0 in the extrusion direction.

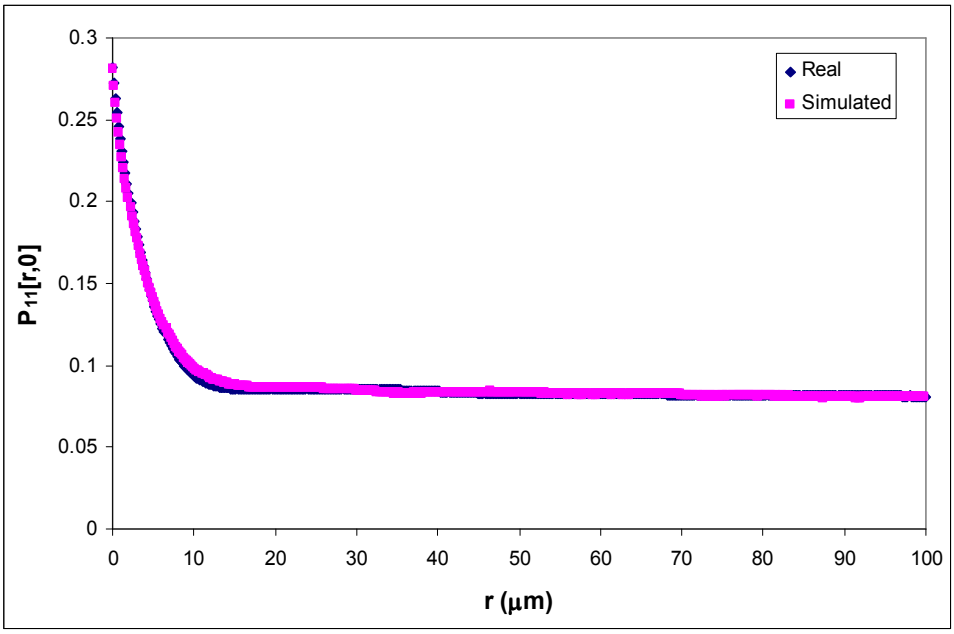



Figure 6.5a: Comparison of two-point distribution functions measured in the extrusion direction for real and simulated microstructures of DRA composites with PSR 3.1

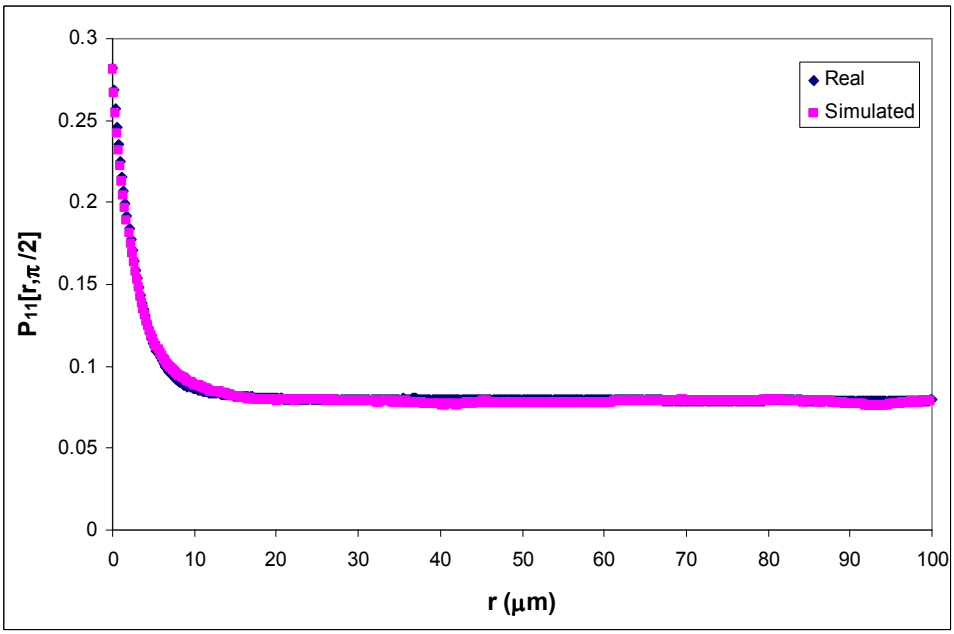



Figure 6.5b: Comparison of two-point distribution functions measured in the transverse direction for real and simulated microstructures of DRA composites with PSR 3.1.

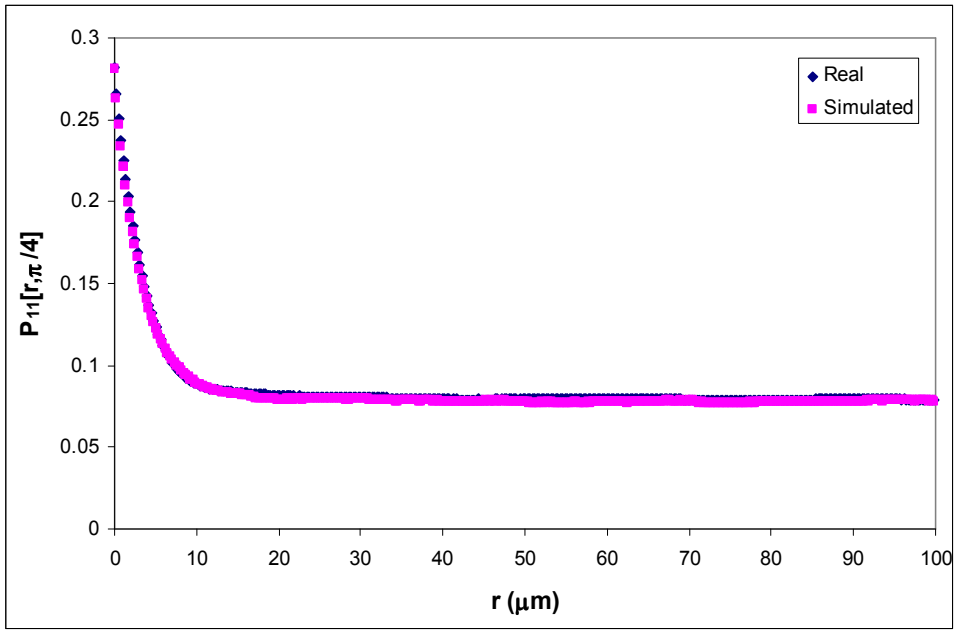



Figure 6.5c: Comparison of two-point distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of DRA composites with PSR 3.1

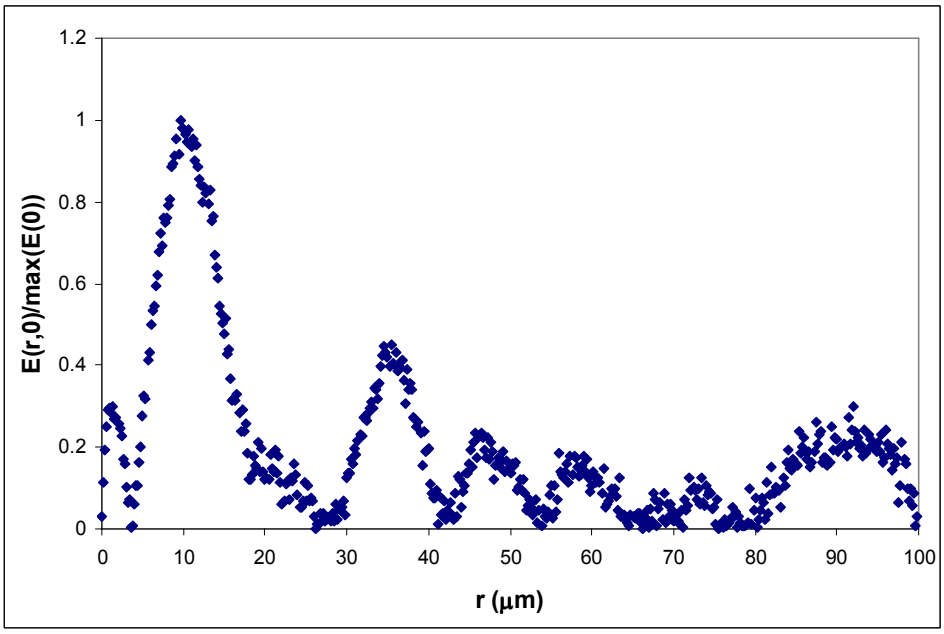



Figure 6.5d: Plot of percentage error between real and simulated two point probability functions normalized by the maximum percentage of 5.52% for PSR 3.1 in the extrusion direction

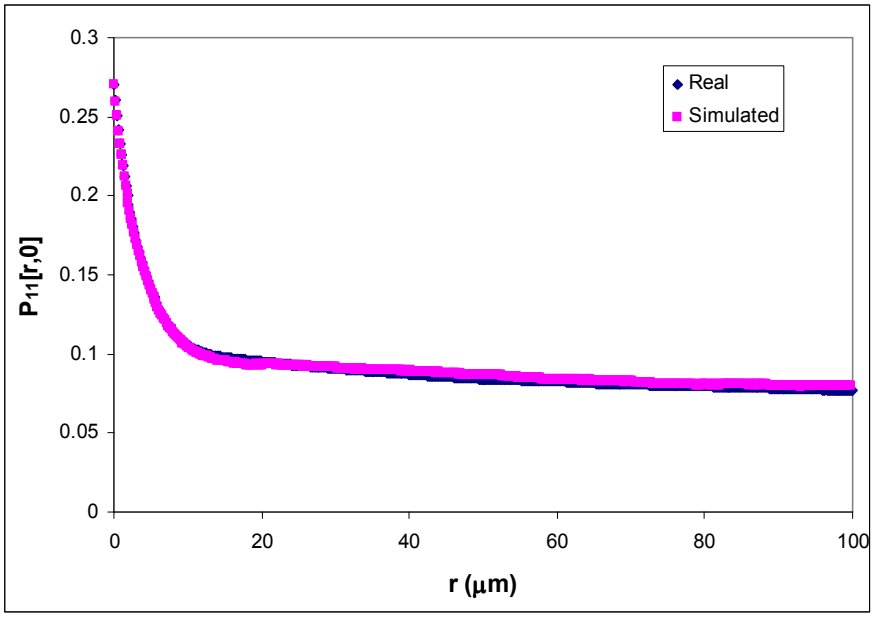



Figure 6.6a: Comparison of two-point distribution functions measured in the extrusion direction for real and simulated microstructures of DRA composites with PSR 8.1

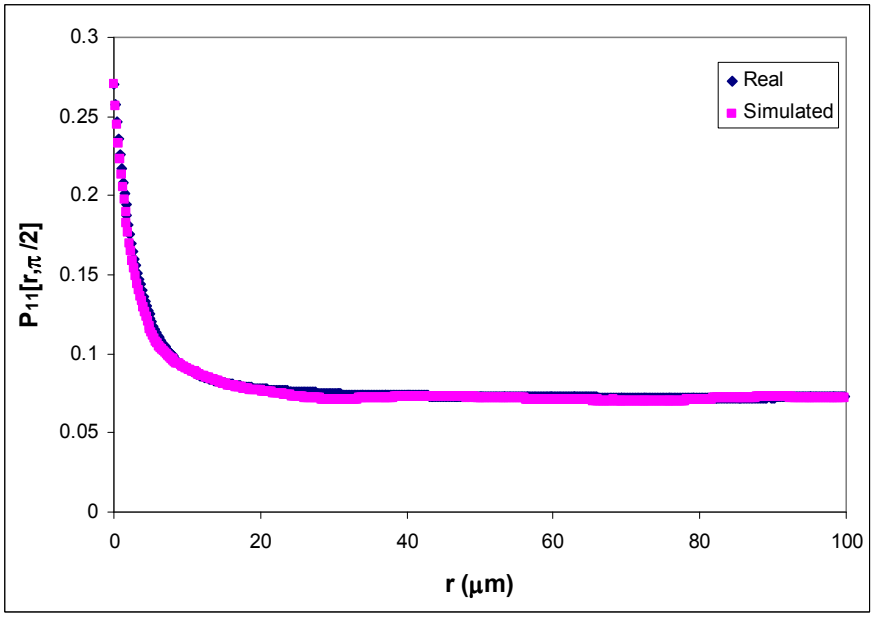



Figure 6.6b: Comparison of two-point distribution functions measured in the transverse direction for real and simulated microstructures of DRA composites with PSR 8.1.

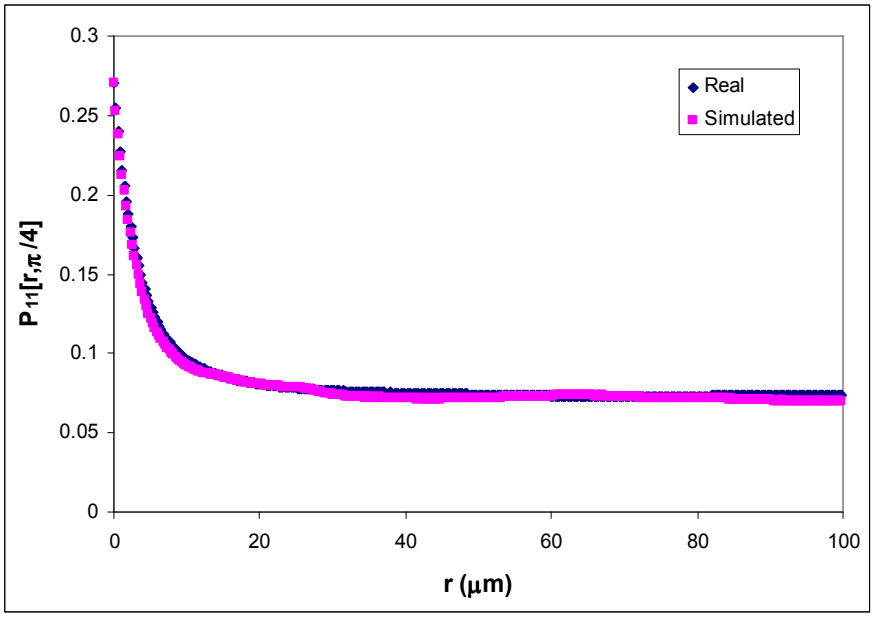



Figure 6.6c: Comparison of two-point distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of DRA composites with PSR 8.1

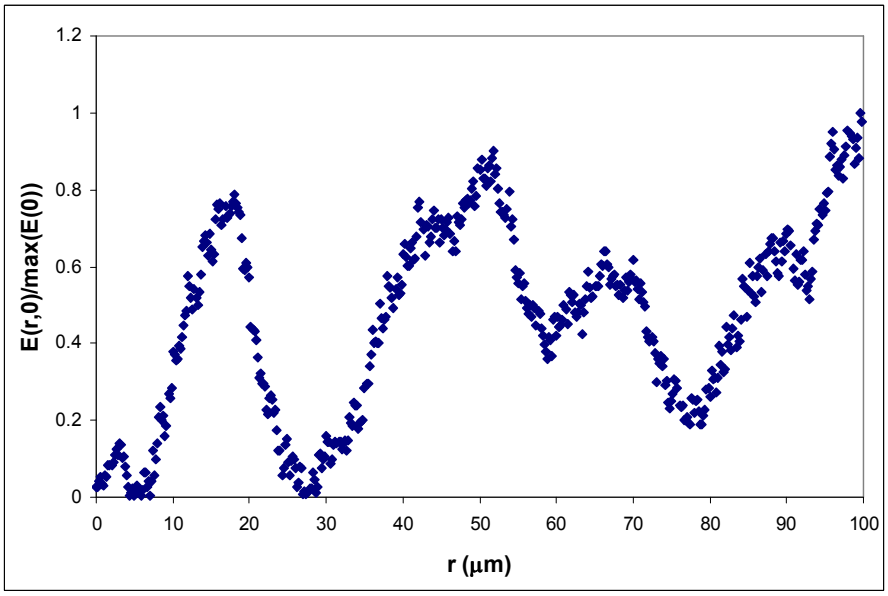



Figure 6.6d: Plot of percentage error between real and simulated two point probability functions normalized by the maximum percentage of 3.83% for PSR 8.1 in the extrusion direction

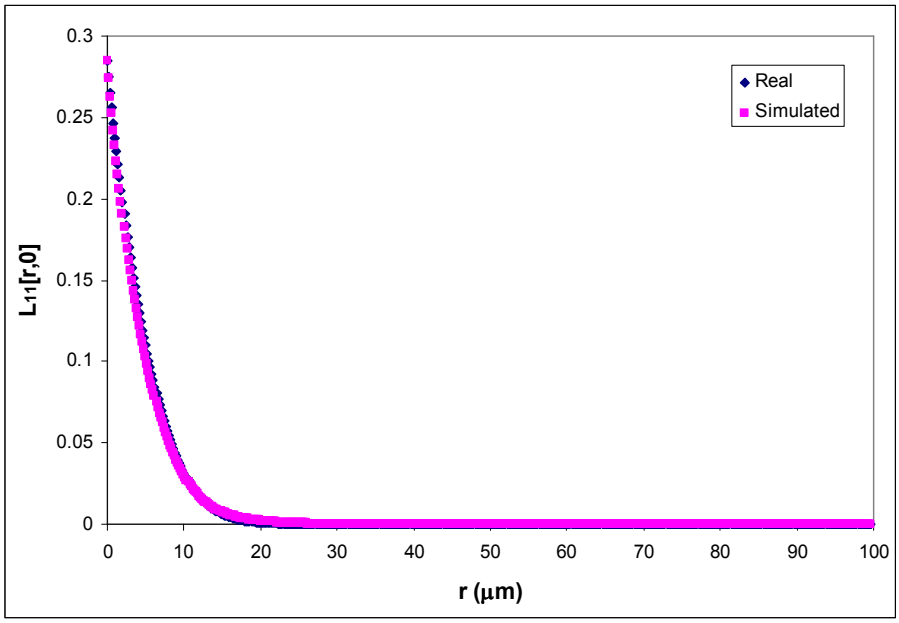



Figure 6.7a: Comparison of lineal path probability functions measured in the extrusion direction for real and simulated microstructures of DRA composites with PSR 2.0

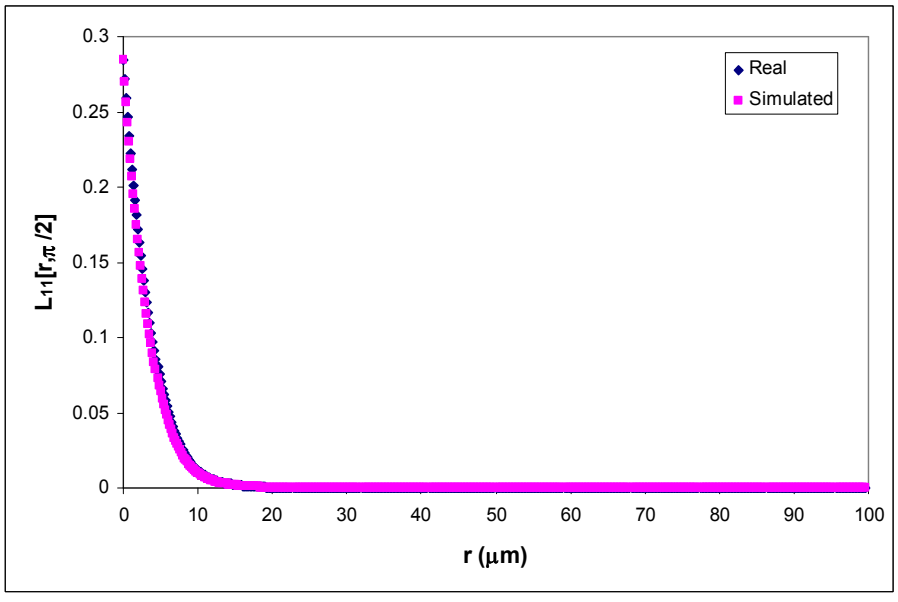



Figure 6.7b: Comparison of lineal path probability functions measured in the transverse direction for real and simulated microstructures of DRA composites with PSR 2.0.

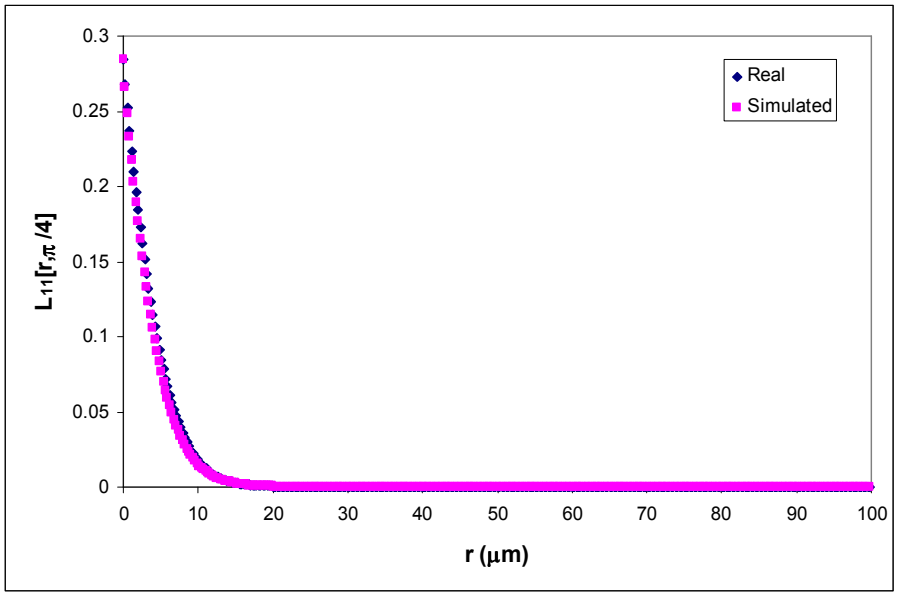



Figure 6.7c: Comparison of lineal path probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of DRA composites with PSR 2.0

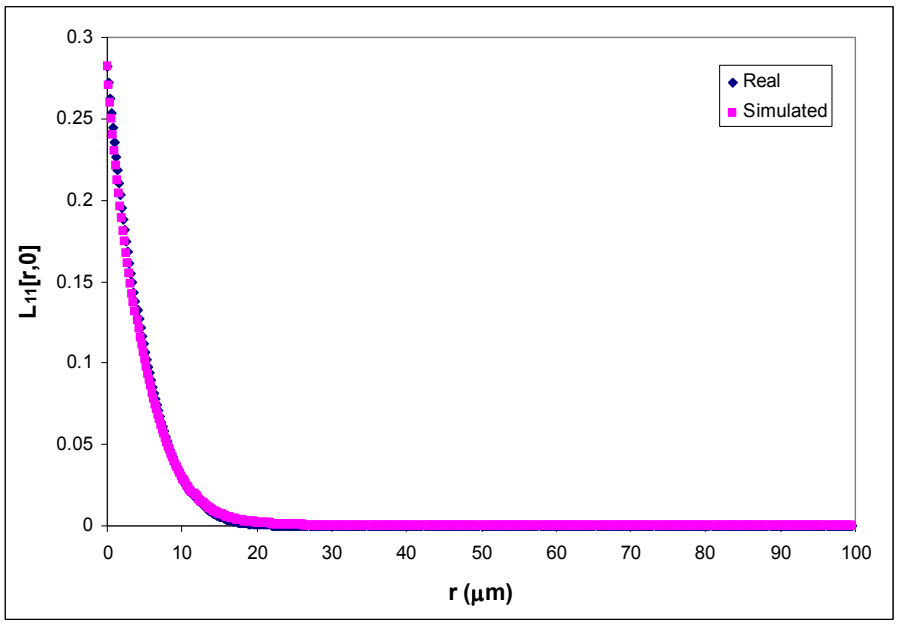



Figure 6.8a: Comparison of lineal path probability functions measured in the extrusion direction for real and simulated microstructures of DRA composites with PSR 3.1

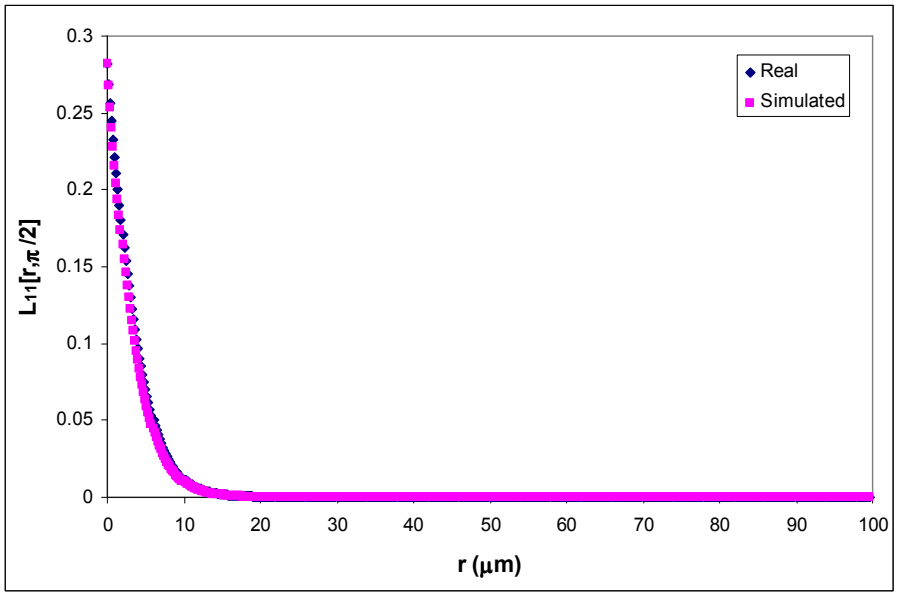



Figure 6.8b: Comparison of lineal path probability functions measured in the transverse direction for real and simulated microstructures of DRA composites with PSR 3.1

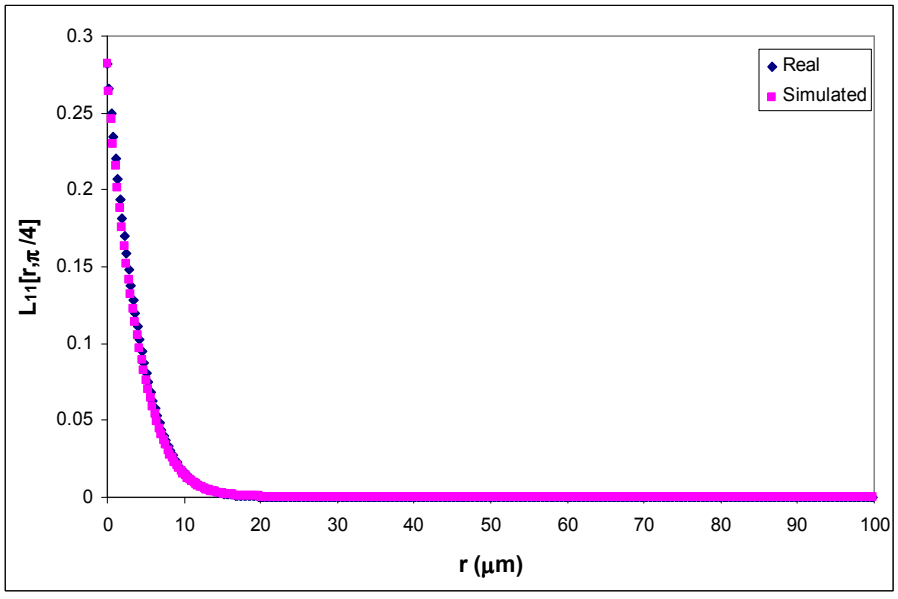



Figure 6.8c: Comparison of lineal path probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of DRA composites with PSR 3.1.

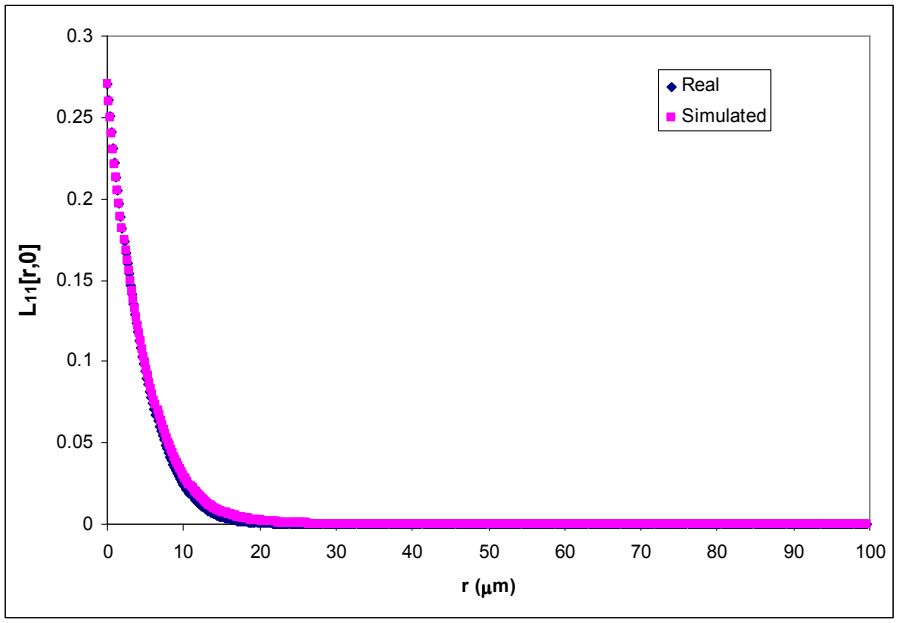



Figure 6.9a: Comparison of lineal path probability functions measured in the extrusion direction for real and simulated microstructures of DRA composites with PSR 8.1

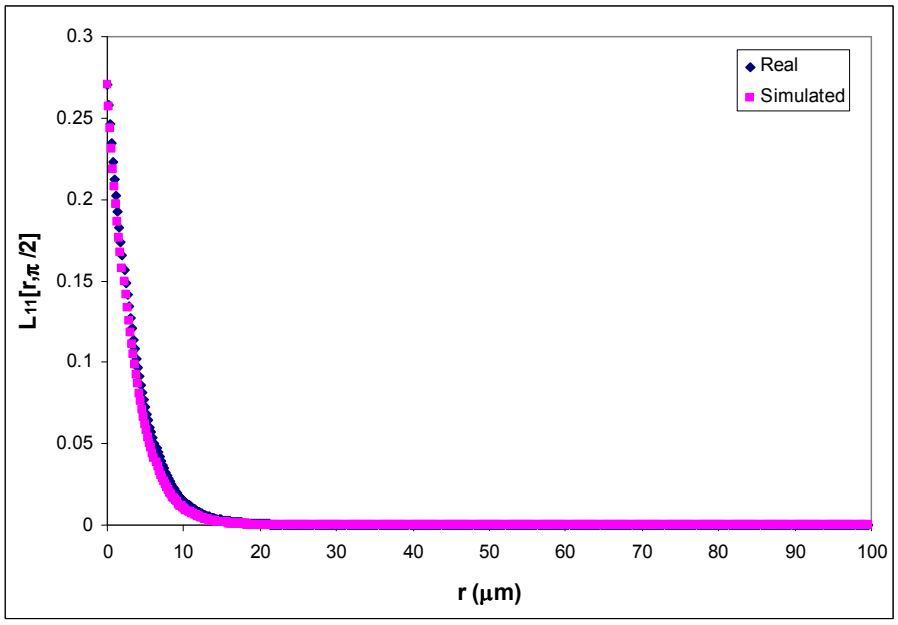



Figure 6.9b: Comparison of lineal path probability functions measured in the transverse direction for real and simulated microstructures of DRA composites with PSR 8.1

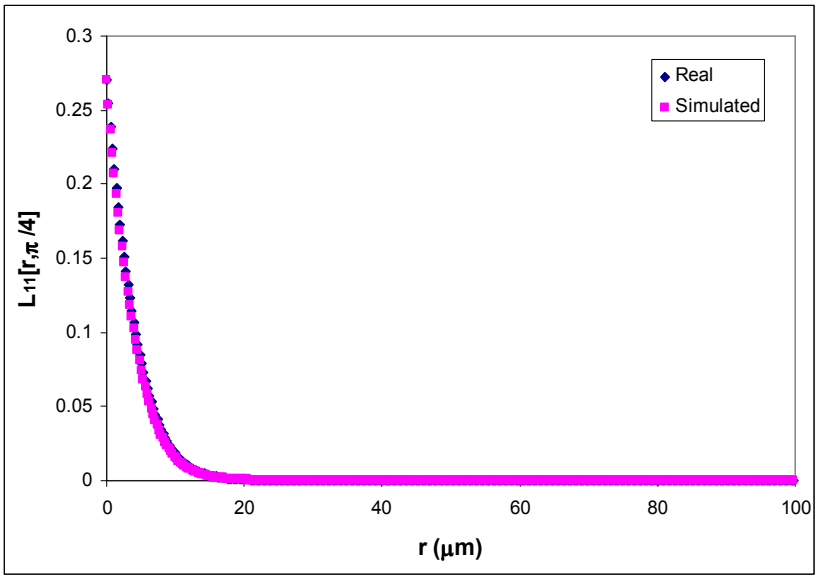



Figure 6.9c: Comparison of lineal path probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of DRA composites with PSR 8.1.

Table 6.1: Values of Simulation parameters used to generate simulated microstructures of DRA composites.

| PSR | Cluster 1 | | | Cluster 2 | | | Particle Overlap | **Cluster Intensity** |
|---|---|---|---|---|---|---|---|---|
| | Number density Per mm$^2$ | Length (μm) | Axial Ratio | Number density per mm$^2$ | Length (μm) | Axial Ratio | | |
| 20 | 36.4 | 350 | 11.67 | 15.6 | 160 | 2.67 | 0.085 | 1.40 |
| 31 | 36.4 | 350 | 11.67 | 15.6 | 160 | 2.67 | 0.085 | 1.52 |
| 81 | 36.4 | 350 | 11.67 | 15.6 | 160 | 2.67 | 0.085 | 1.88 |

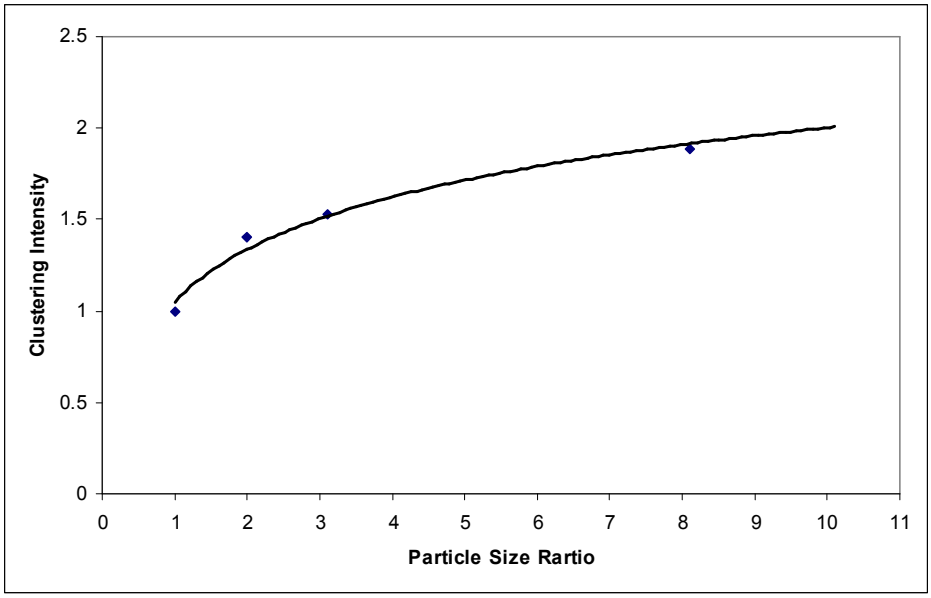



Figure 6.10: Relationship between particle size ratio (processing parameter) and clustering intensity (simulation parameter)

### 6.2. Simulation of Virtual Microstructures of DRA Composites

The ultimate goal of this methodology is to be able to generate a library of microstructures that correspond to different processing conditions starting from a few specimens. These microstructures created as a consequence of the simulation of realistic microstructures are termed as "virtual" microstructures and they mimic the change in the processing parameters by changing the simulation parameters. The input required for generating such virtual microstructures is the set of real particles/features representing the complete size/shape distribution of particles/features in the corresponding real microstructures and the set of simulation parameters obtained by creating realistic simulations of microstructures of the given specimens representing different processing conditions. For the present DRA specimens, the correlation between PSR (a process parameter) and clustering intensity (a simulation parameter) is shown in the Figure 6.10.

This correlation has been utilized to simulate the virtual microstructures corresponding to different PSR values that have the same volume fraction and size and shape distribution of SiC particles as in the real microstructures. Such simulations would represent a set of microstructures where all process parameters except PSR have been kept constant. Figure 6.11 shows an example of such a microstructure for a composite with PSR 6.0. This virtual microstructure has been generated by using the correlation in Figure 6.10, which indicates that the clustering intensity for the composite having 6.0 PSR should be 1.78. The virtual microstructure has been simulated with all details using the same simulation model but changing the clustering intensity value to 1.78. The microstructure in Figure 6.11 is a virtual microstructure because it was generated without actual fabrication of the corresponding composite. Note that this virtual microstructure has the same realistic SiC particle morphologies[2], and it incorporates realistic short-range (0 to 10 μm), intermediate-range (10 to 50 μm), and long-range (50 to 500 μm) spatial patterns and microstructural details at high resolution (0.2 μm pixel size). Similarly, one can simulate virtual microstructures of composites that cover a complete range of PSR values as illustrated in Figures 12 and 13, which show the virtual microstructures with PSR 1.0 and 10.0 respectively.

The simulation parameters can also be utilized to generate virtual microstructures of DRA composites with different volume fraction of SiC particles than in the three given specimens via this methodology. Figures 6.14 to 6.19 shows a set of virtual microstructure for composites with PSR 1, 2.0 3.1, 6.0, 8 and 10 each having a different volume fraction of SiC particles of 10% and 20%, which is lower than the volume

[2] Note that changing PSR or volume fraction of SiC particles does not change SiC particle shapes/morphologies and size distribution if the same SiC powder population is used for the fabrication of the composite.

fraction in the real specimens. Further, virtual microstructures of DRA composites with different particle size distribution than in the given specimens can also be generated. Figure 6.20 shows a virtual microstructure with lower average size of the SiC particles than in the real composites.

Furthermore, since the shape of the clusters in the simulation model represents the extent of extrusion process on the specimens, virtual microstructures depicting different level of extrusion ratios can also be generated by using the same set of input SiC particles and varying the cluster parameters in the simulation model. Figures 6.21a and 6.21b show an example of virtual microstructures with PSR 8.1 but with lower and higher extrusion ratios as compared to the corresponding real microstructure, respectively. These virtual microstructures can be then incorporated in the FE-based computational models to predict the mechanical response (and properties) of such virtual materials. The resulting data can provide useful information for materials by design, and the methodology can cut down on the number of experiments (and therefore, time and resources required) for developing new composites and for optimizing the properties of the existing composites. Figure 6.22 shows such an example where stress-strain curve has been plotted for virtual microstructure with PSR 6.0 by incorporating it into the FE-based computational model. The computations for FE based analysis have been performed by Arun Sreeranganathan.

Since the shape of the SiC particles is equiaxed, in the above simulations the orientation of the SiC particles was not changed while placing them in the simulation space. On the hand, in numerous microstructures the constituent phases have preferred morphological orientations leading to partially or completely anisotropic microstructures. The methodology can also be used to generate virtual microstructures with different

degrees of morphological anisotropy, which is discussed in the following section by the simulation of boron whiskers in the boron modified Ti-alloys.

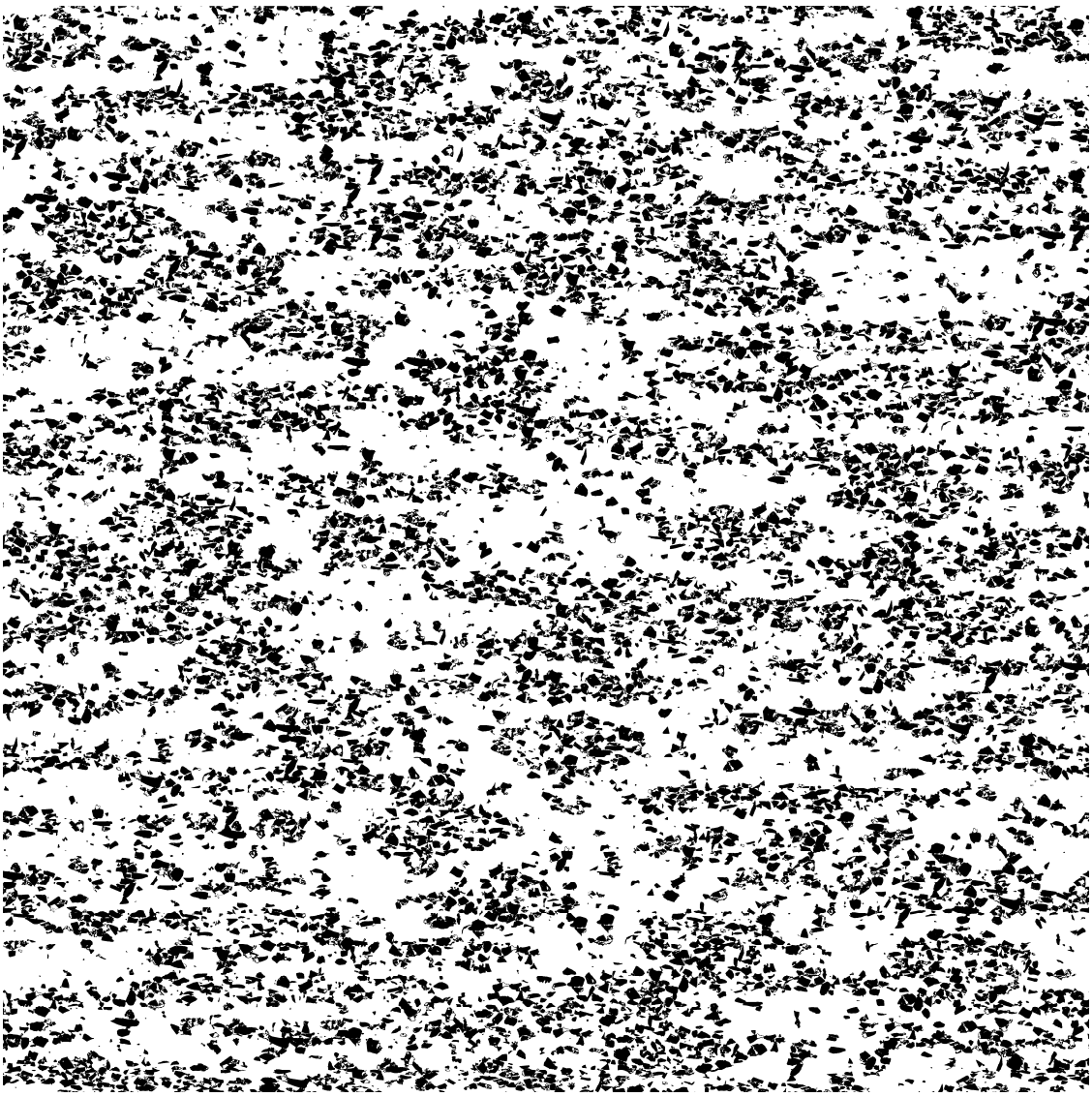


Figure 6.11: Simulated image for virtual microstructure with PSR = 6.0 and volume fraction of SiC particles of 28%.

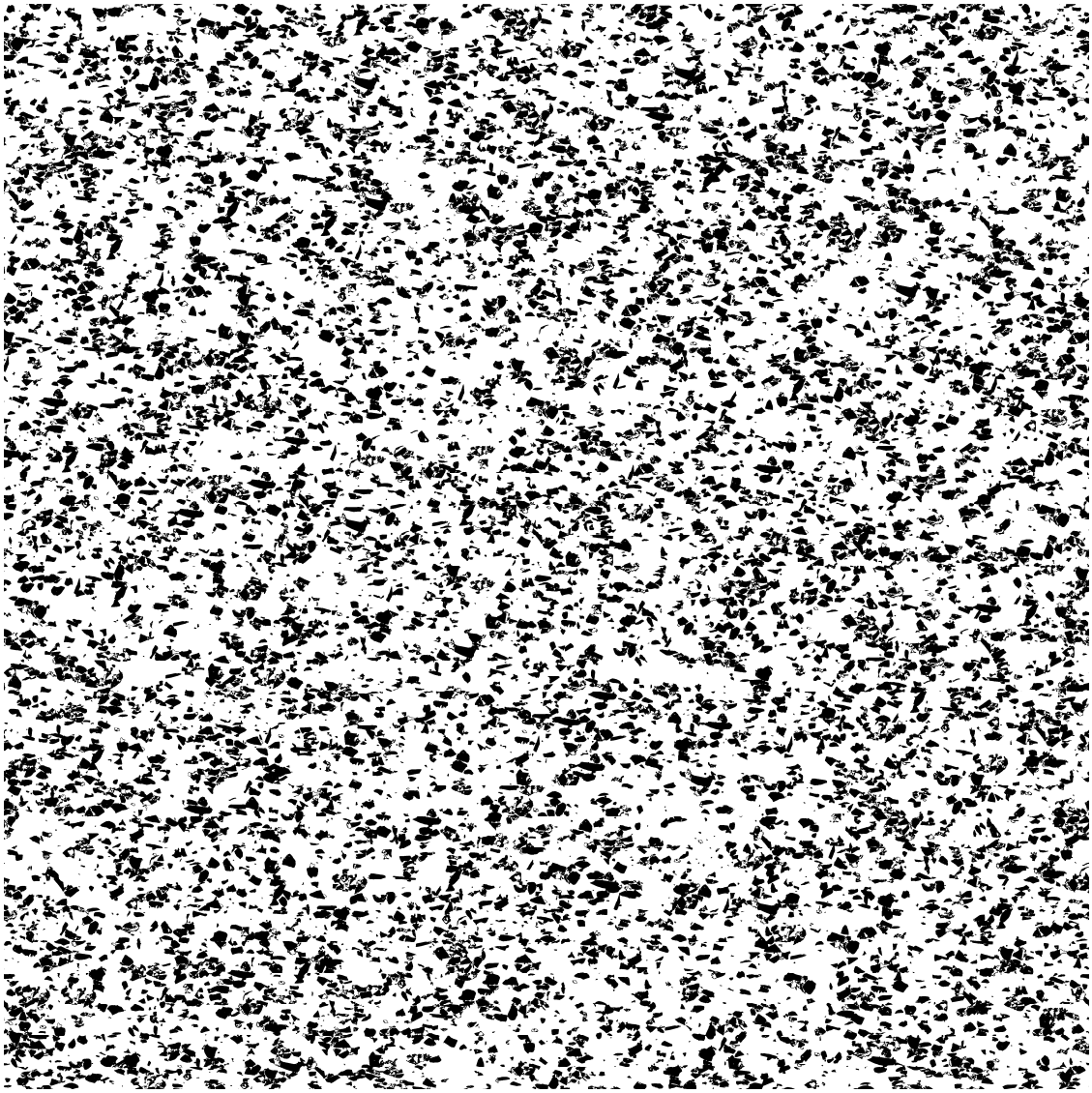


Figure 6.12: Simulated image for virtual microstructure with PSR = 1.0 and volume fraction of SiC particles of 28%.

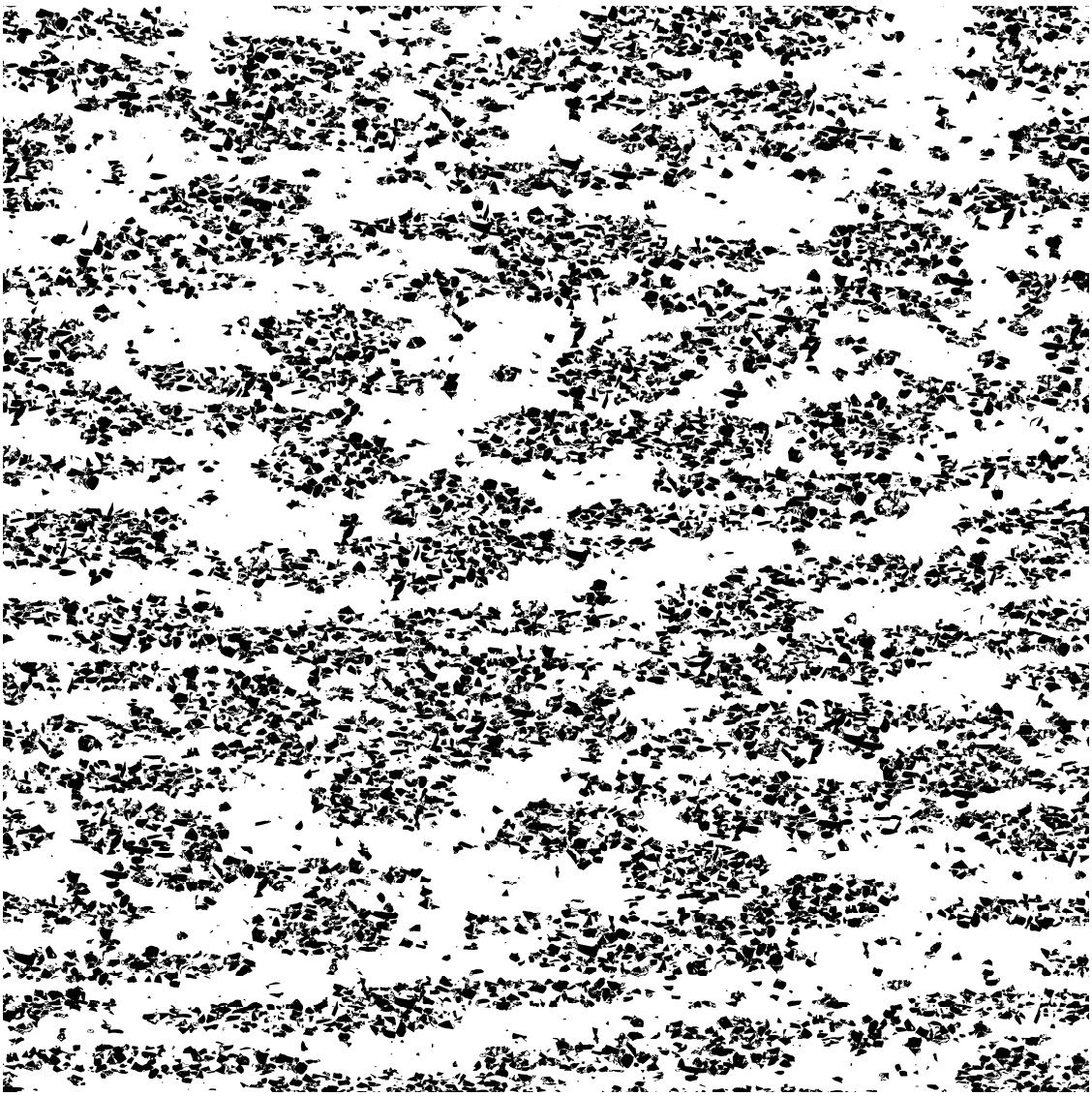


Figure 6.13: Simulated image for virtual microstructure with PSR = 10.0 and volume fraction of SiC particles of 28%.

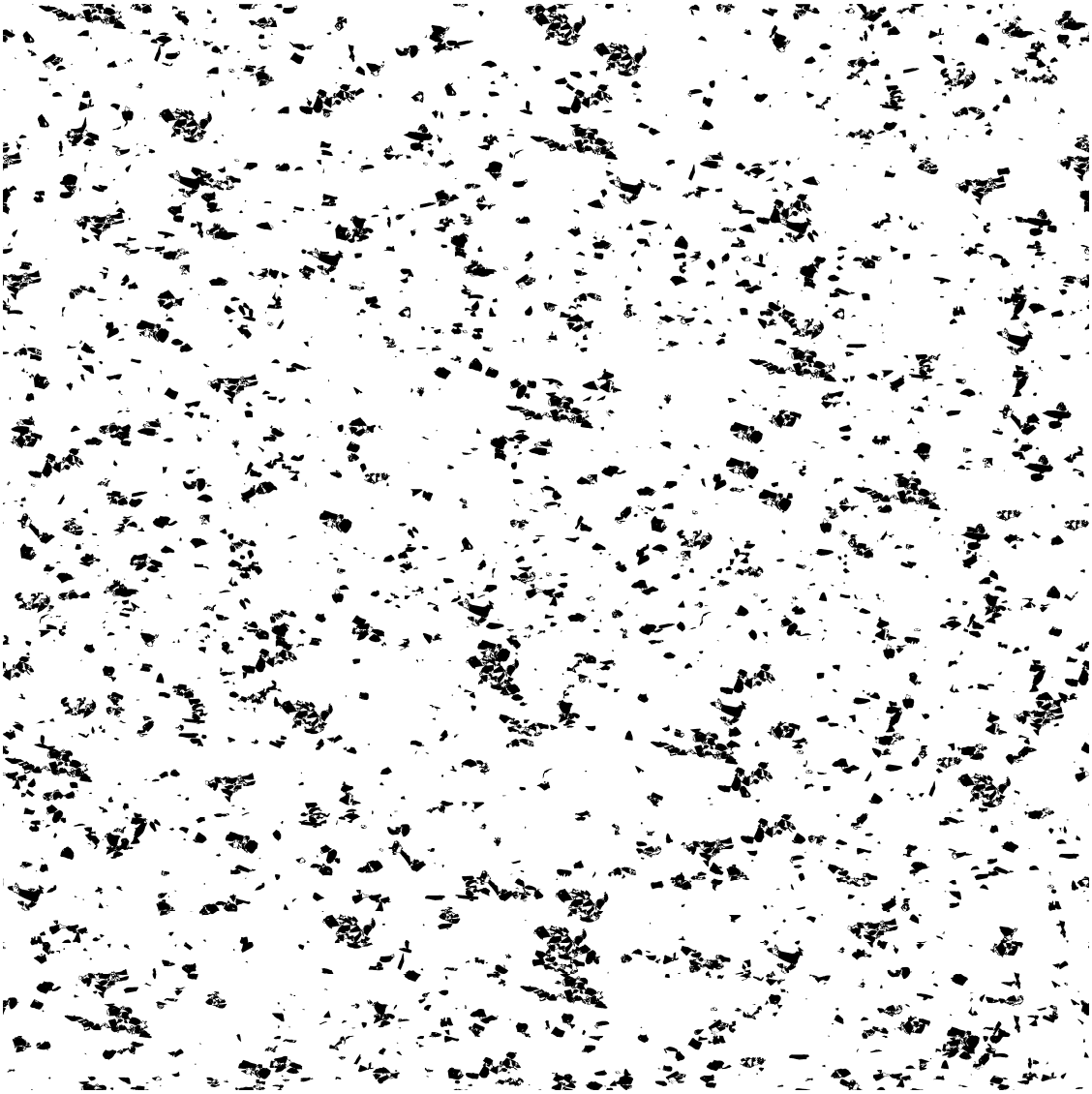


Figure 6.14a: Simulated image for virtual microstructure with PSR = 1.0 and volume fraction of SiC particles of 10%.

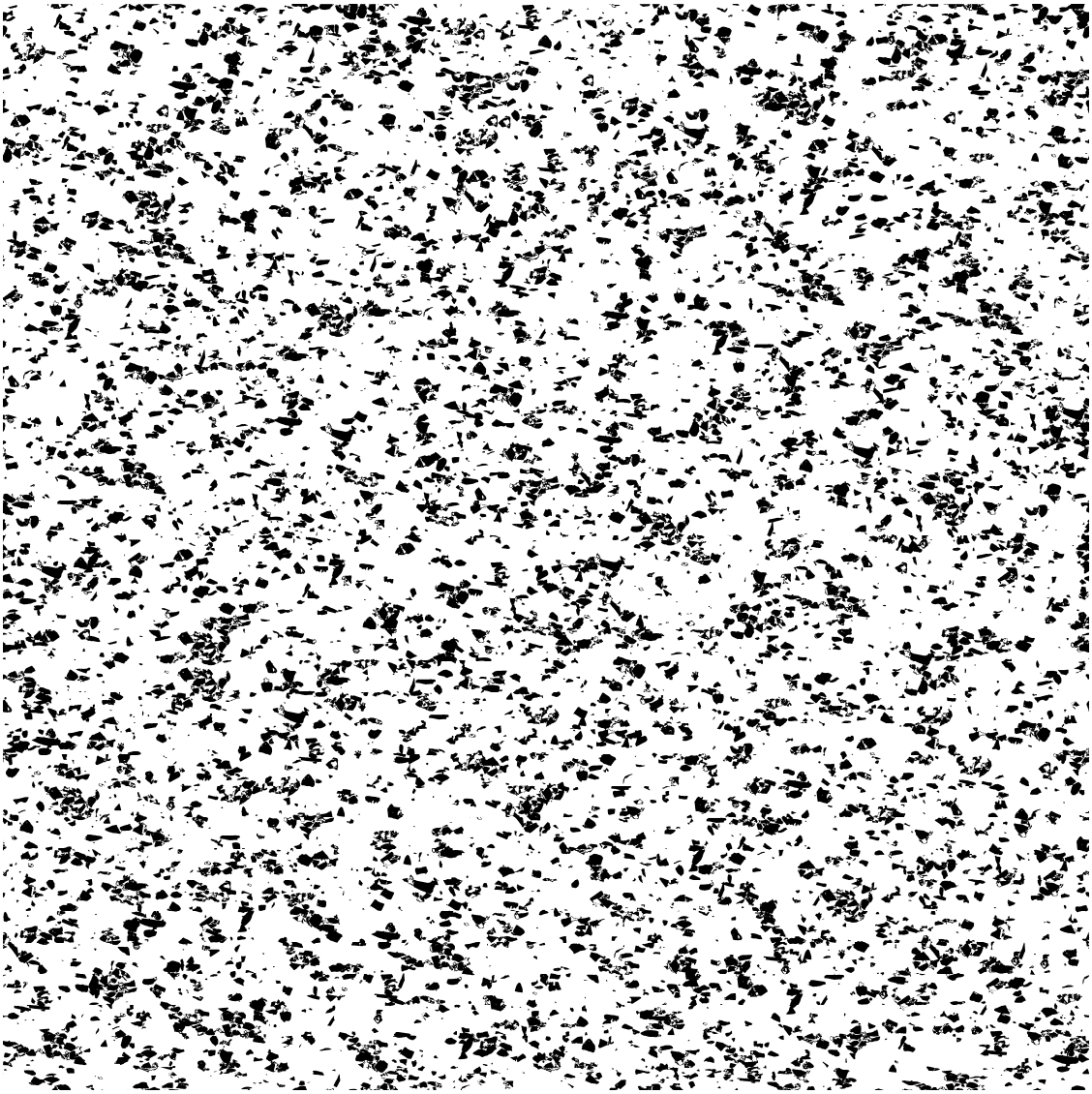


Figure 6.14b: Simulated image for virtual microstructure with PSR = 1.0 and volume fraction of SiC particles of 20%.

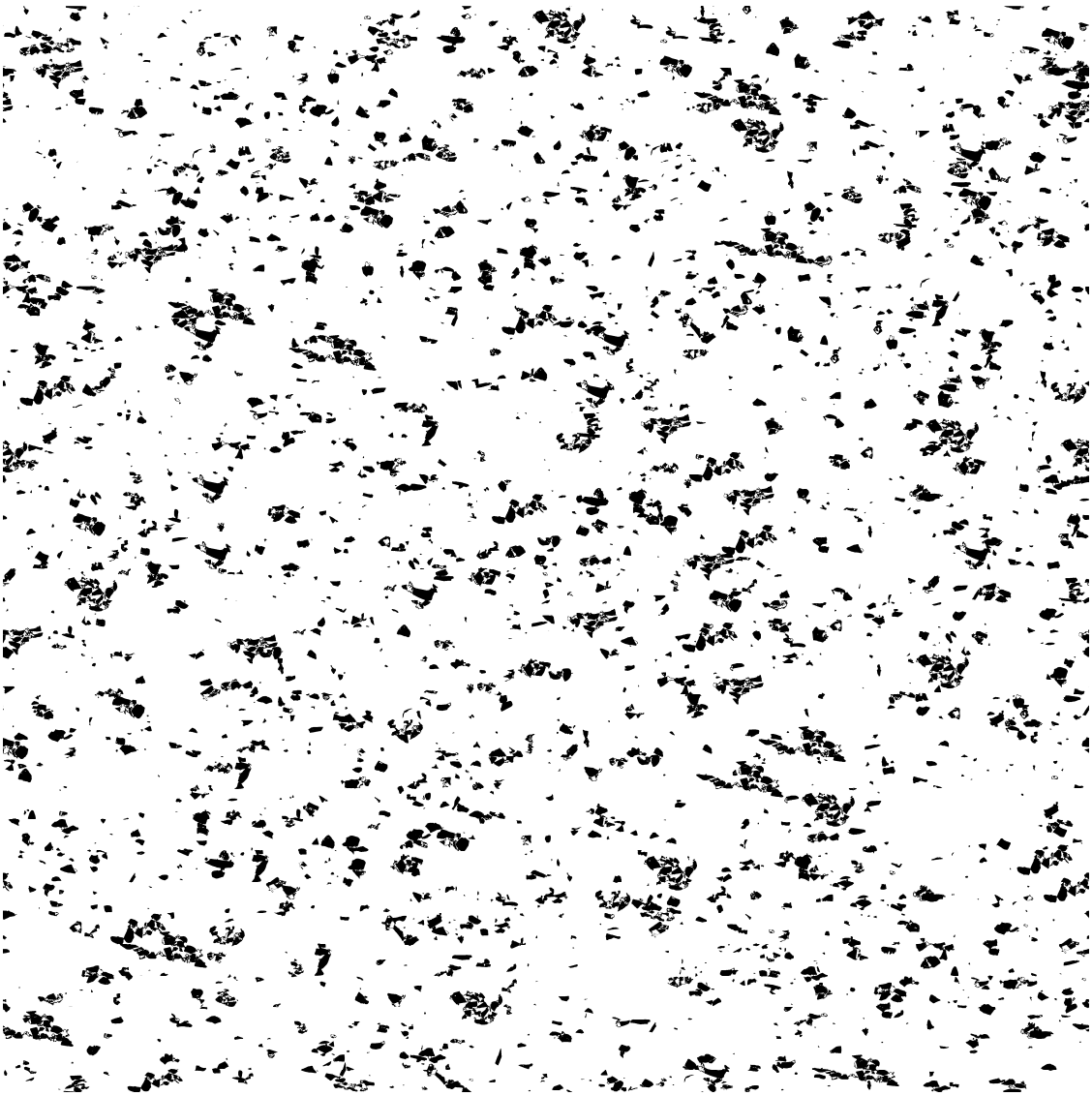


Figure 6.15a: Simulated image for virtual microstructure with PSR = 2.0 and volume fraction of SiC particles of 10%.

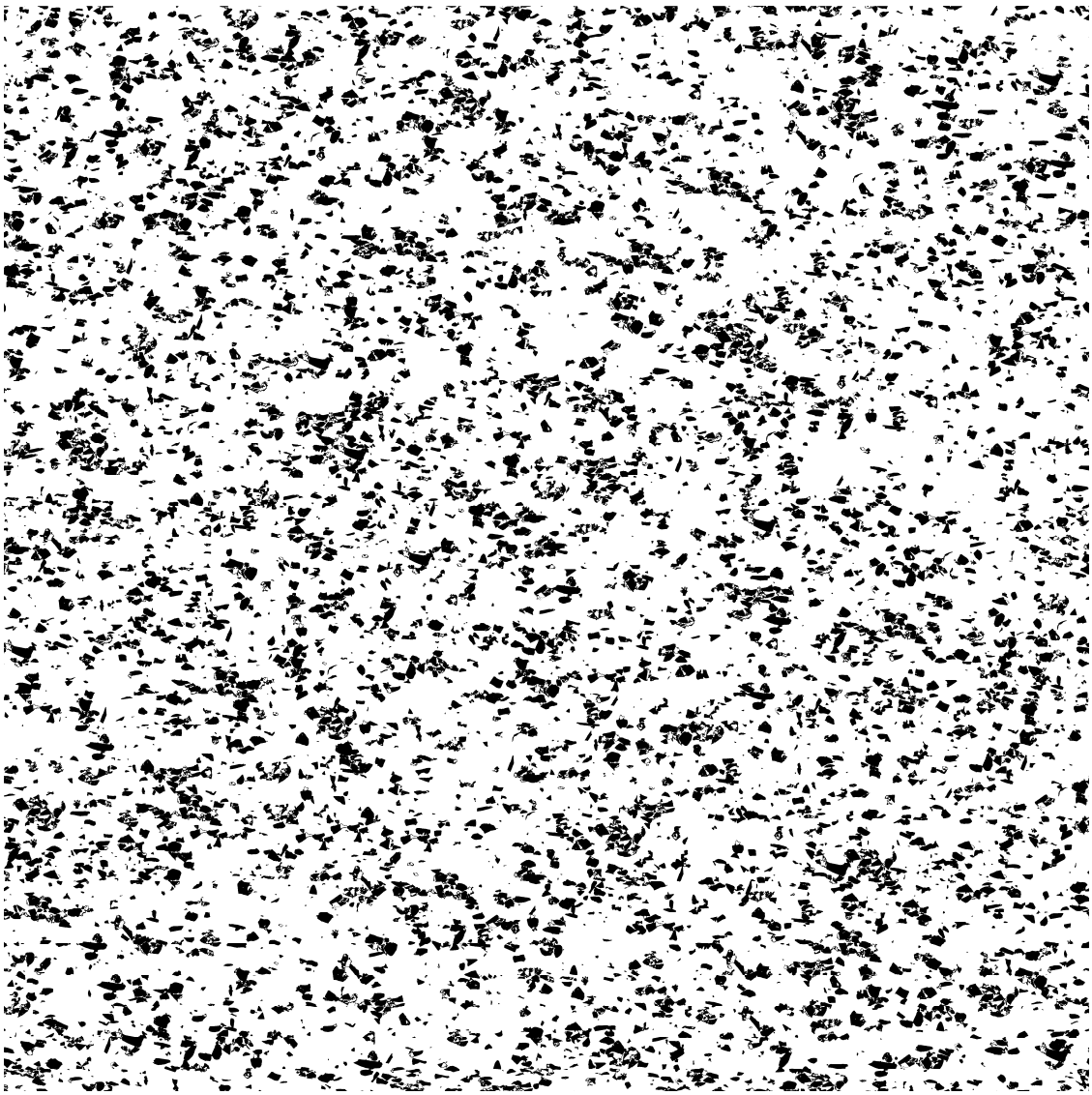


Figure 6.15b: Simulated image for virtual microstructure with PSR = 2.0 and volume fraction of SiC particles of 20%.

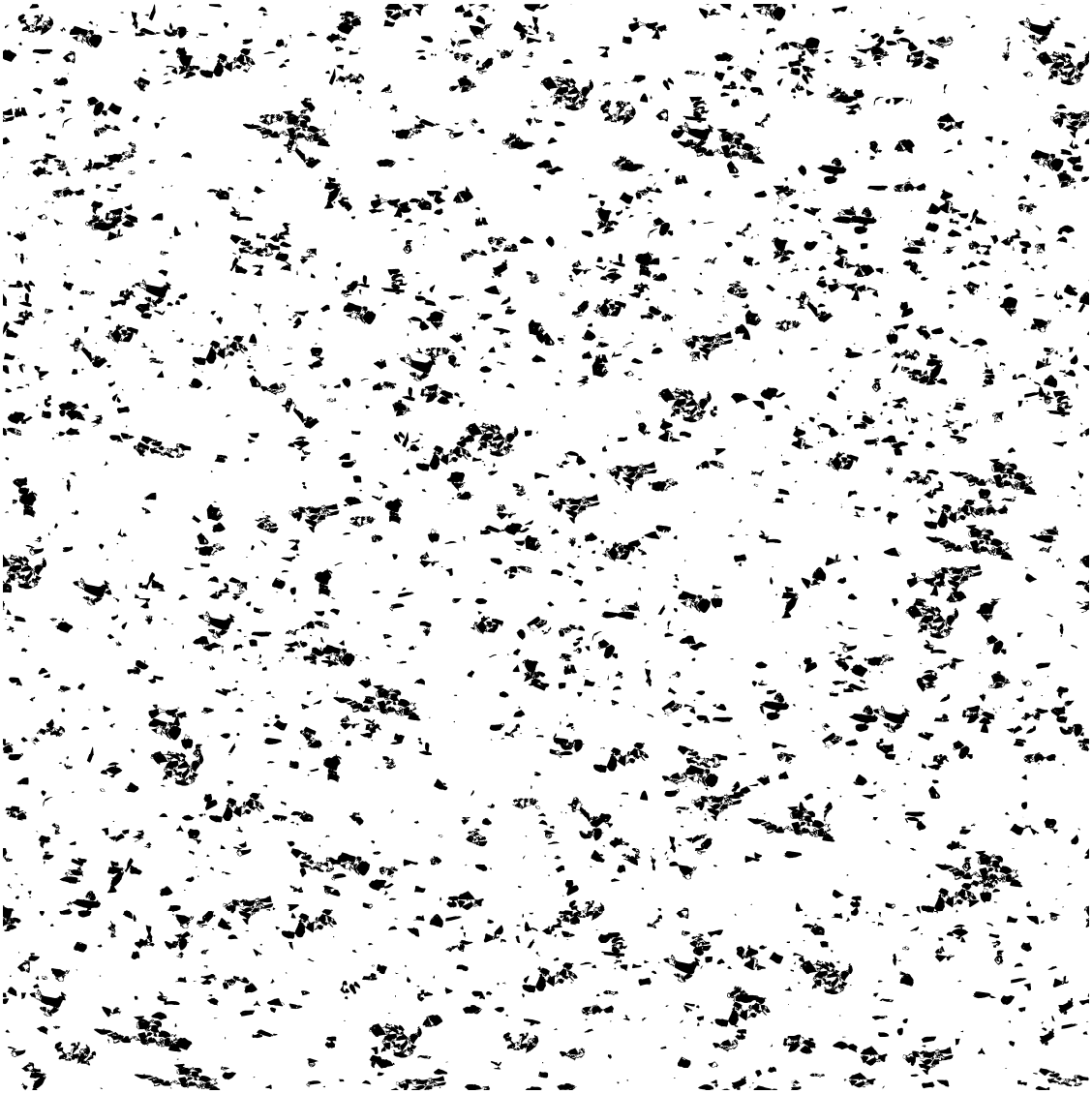


Figure 6.16a: Simulated image for virtual microstructure with PSR = 3.1 and volume fraction of SiC particles of 10%.

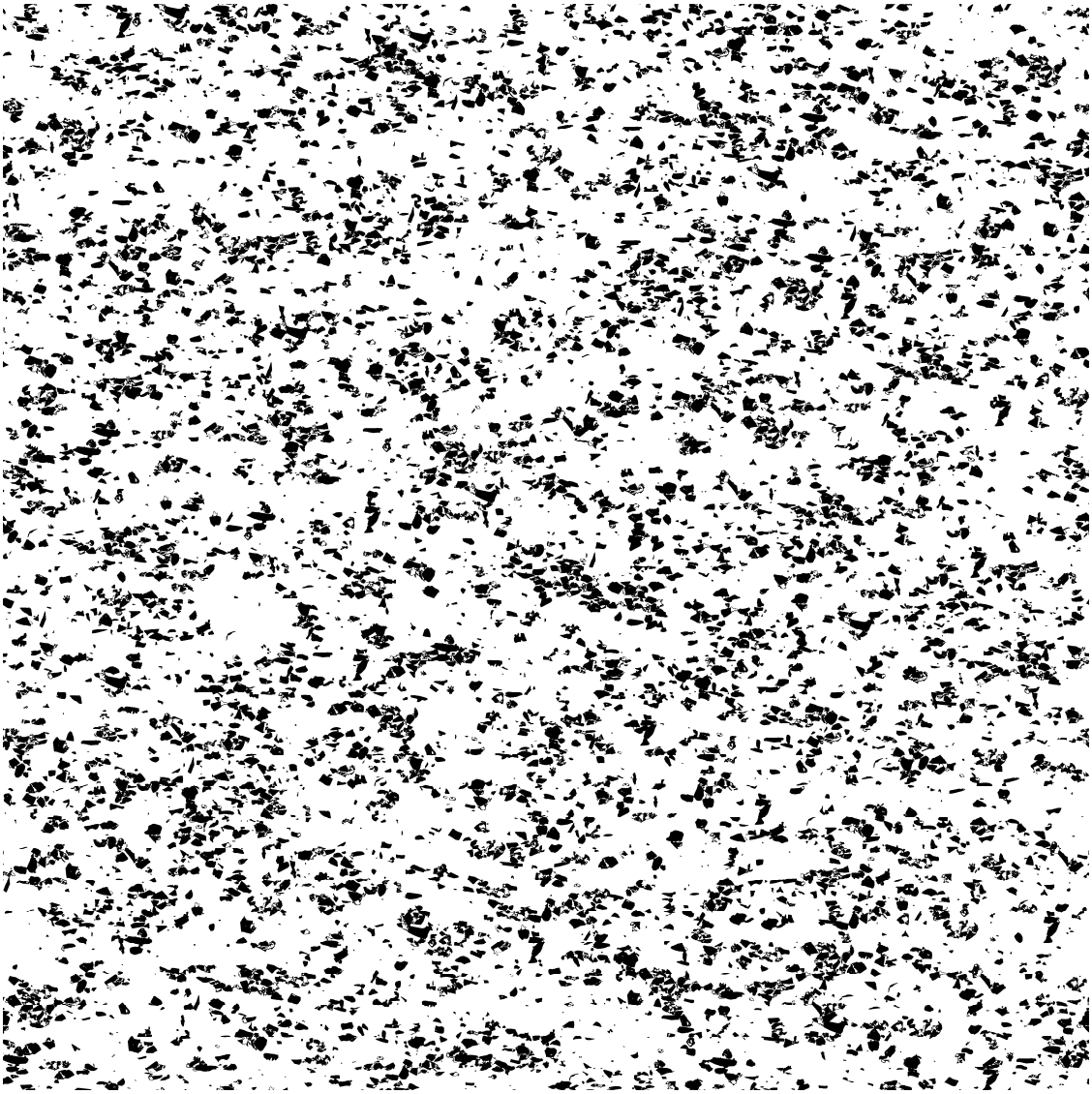


Figure 6.16b: Simulated image for virtual microstructure with PSR = 3.1 and volume fraction of SiC particles of 20%.

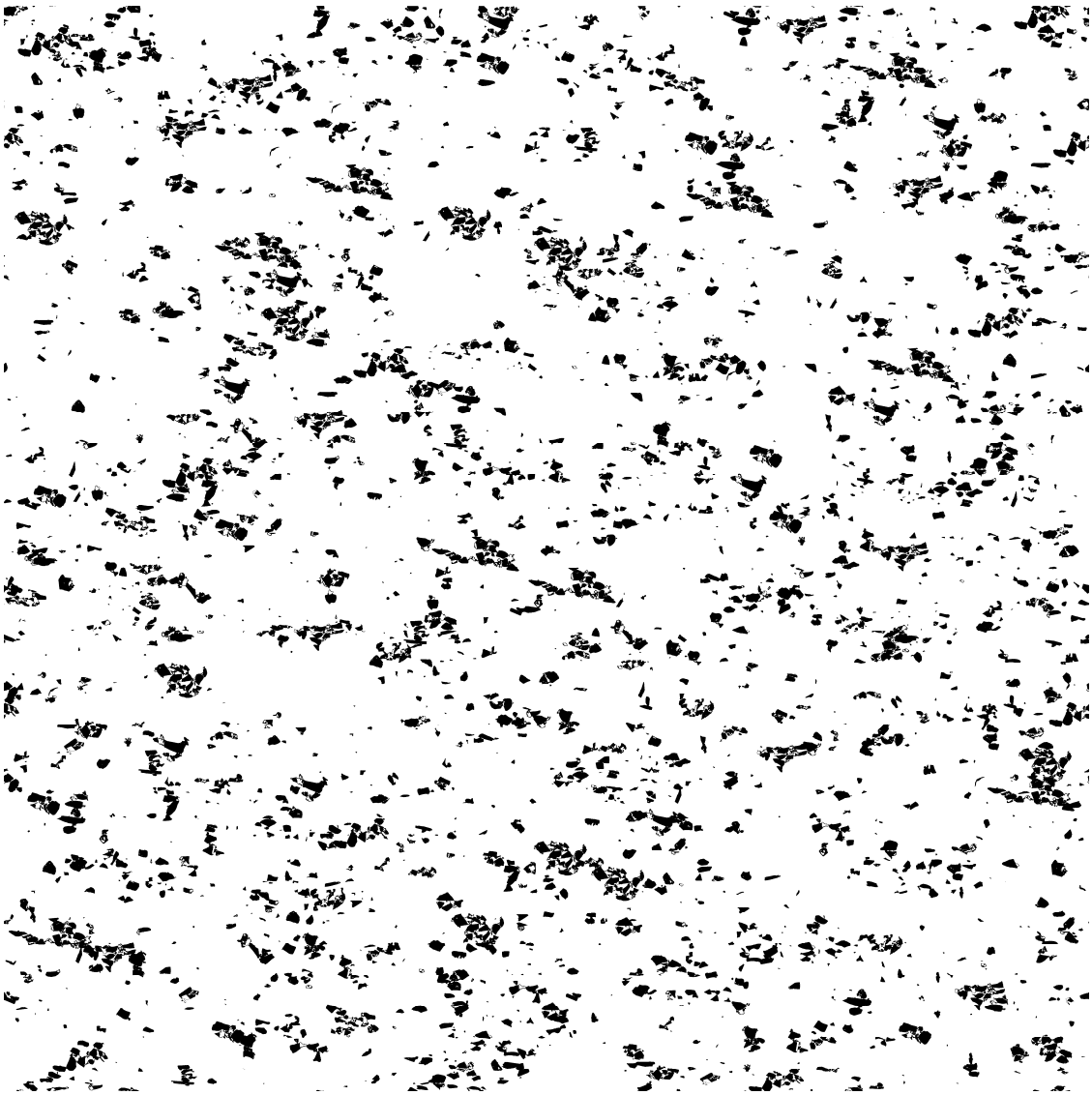


Figure 6.17a: Simulated image for virtual microstructure with PSR = 6.0 and volume fraction of SiC particles of 10%.

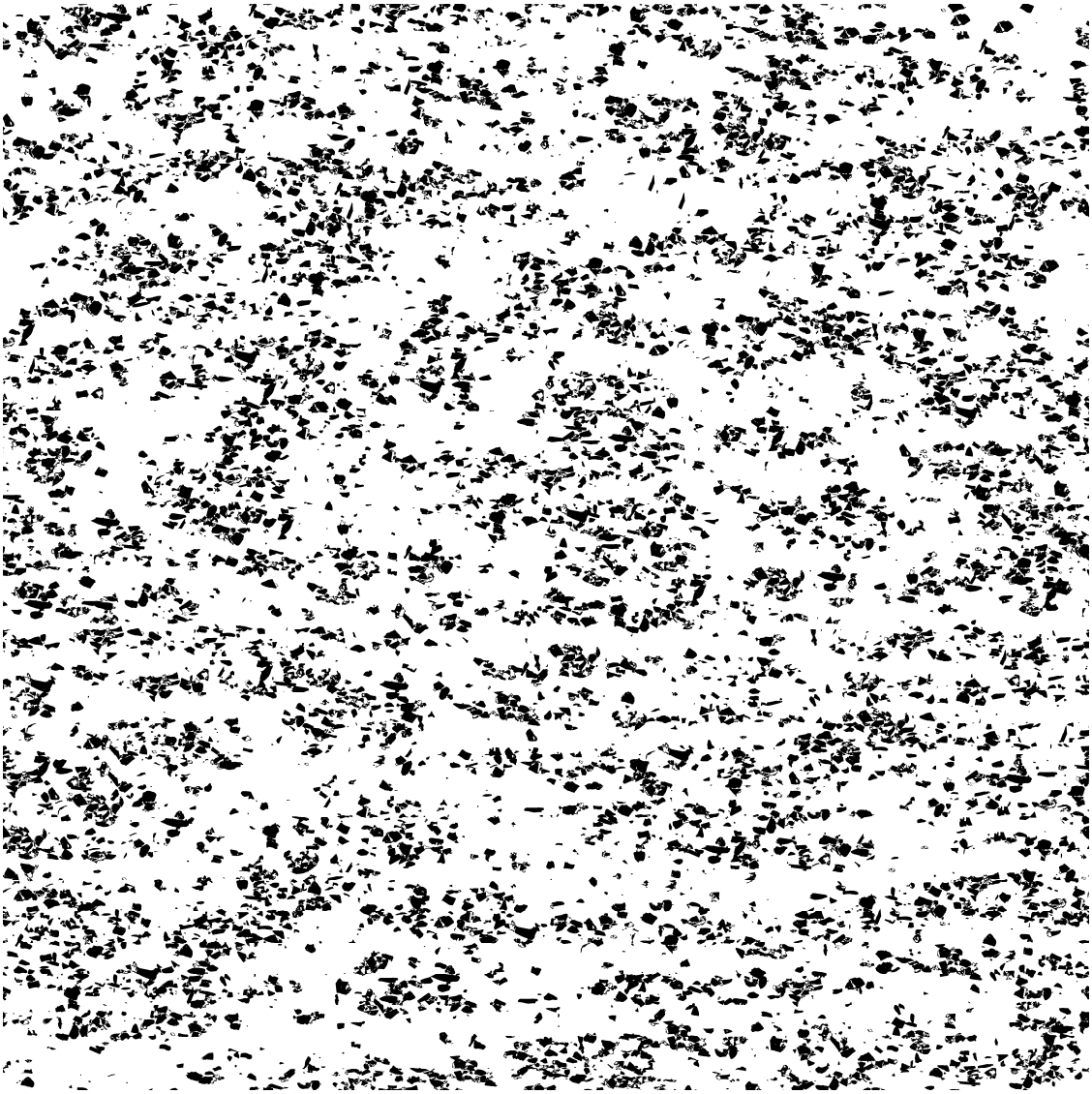


Figure 6.17b: Simulated image for virtual microstructure with PSR = 6.0 and volume fraction of SiC particles of 20%.

Figure 6.18a: Simulated image for virtual microstructure with PSR = 8.1 and volume fraction of SiC particles of 10%.

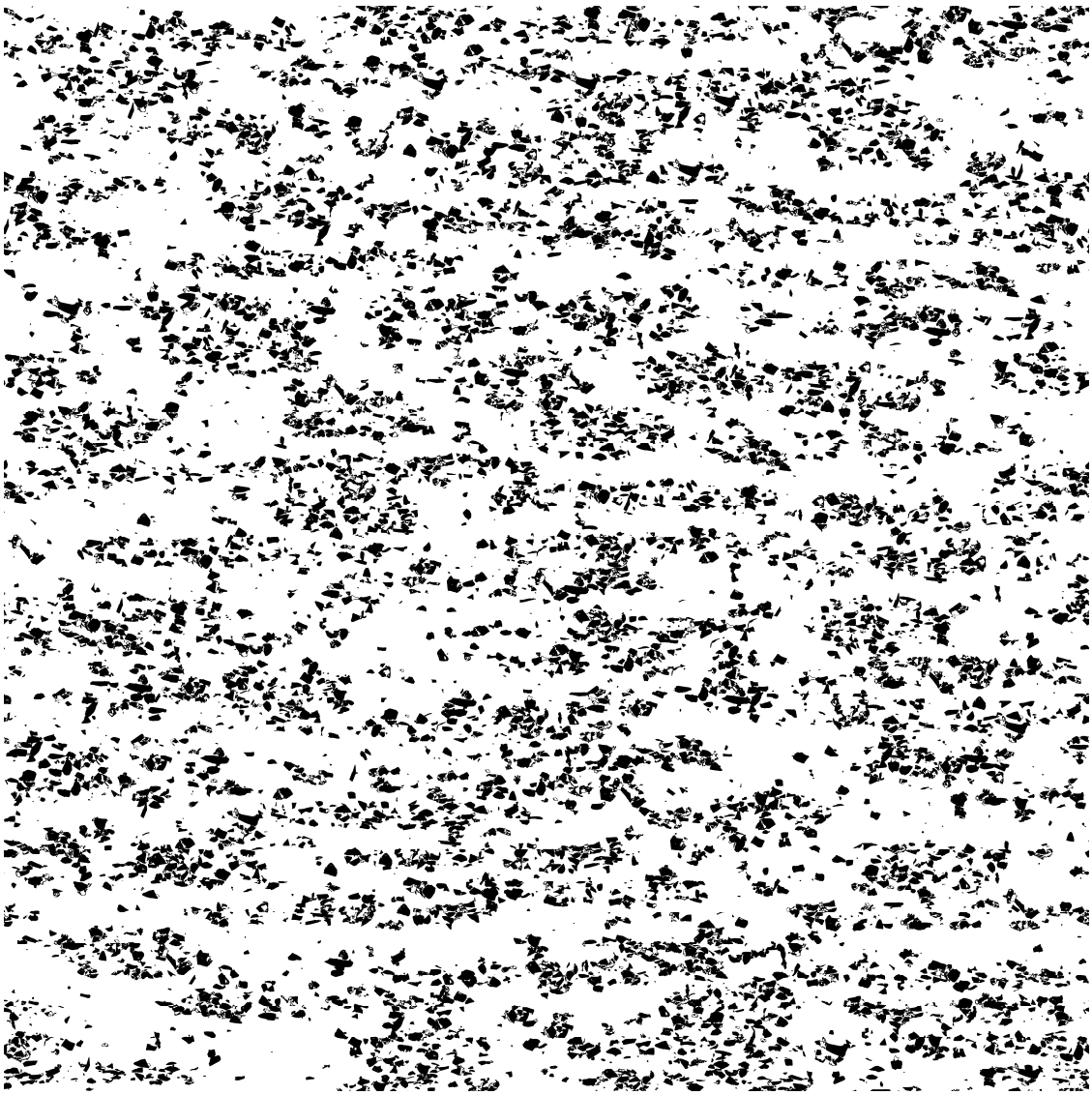


Figure 6.18b: Simulated image for virtual microstructure with PSR = 8.1 and volume fraction of SiC particles of 20%.

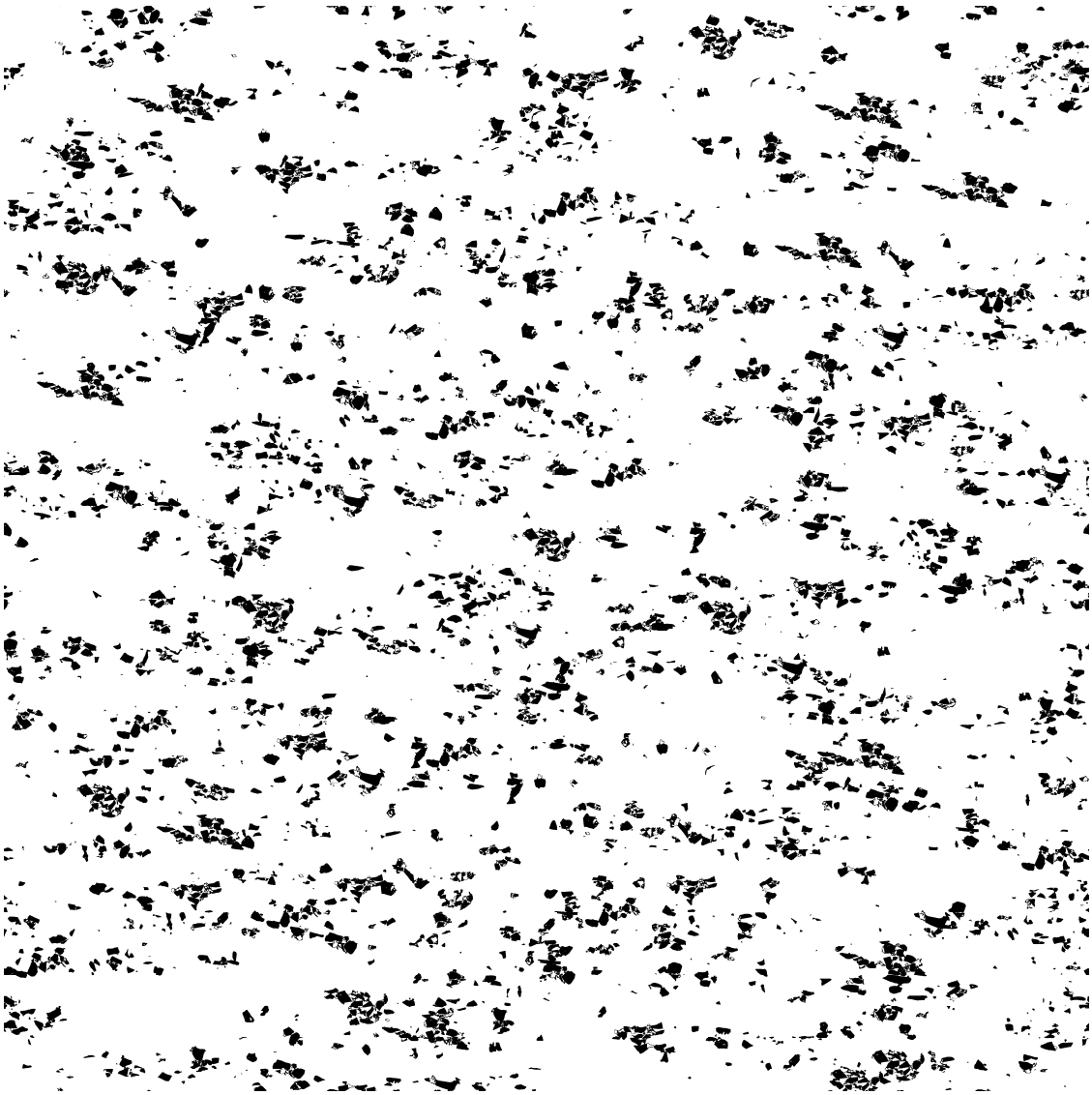


Figure 6.19a: Simulated image for virtual microstructure with PSR = 10.0 and volume fraction of SiC particles of 10%.

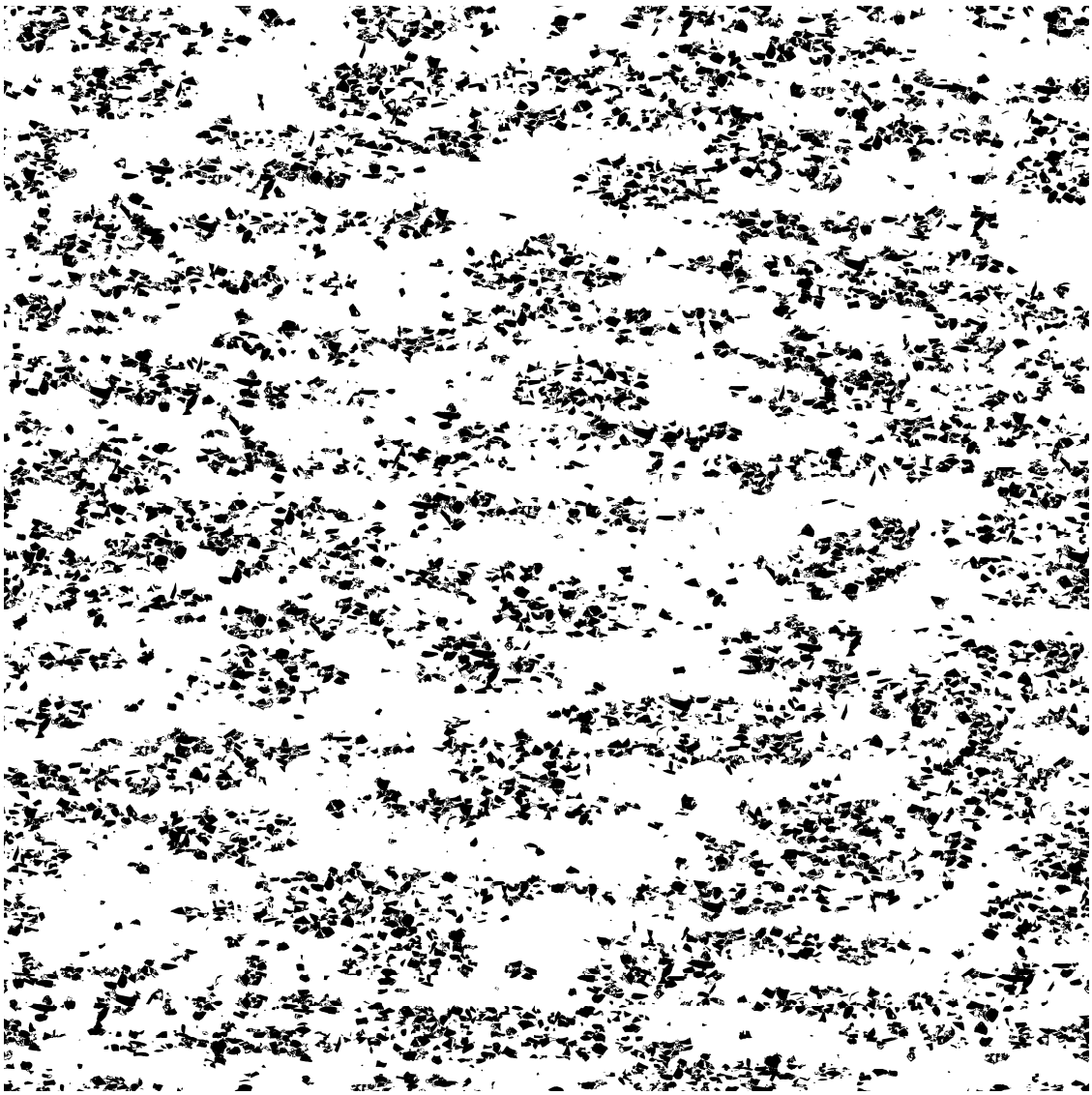


Figure 6.19b: Simulated image for virtual microstructure with PSR = 10.0 and volume fraction of SiC particles of 20%.

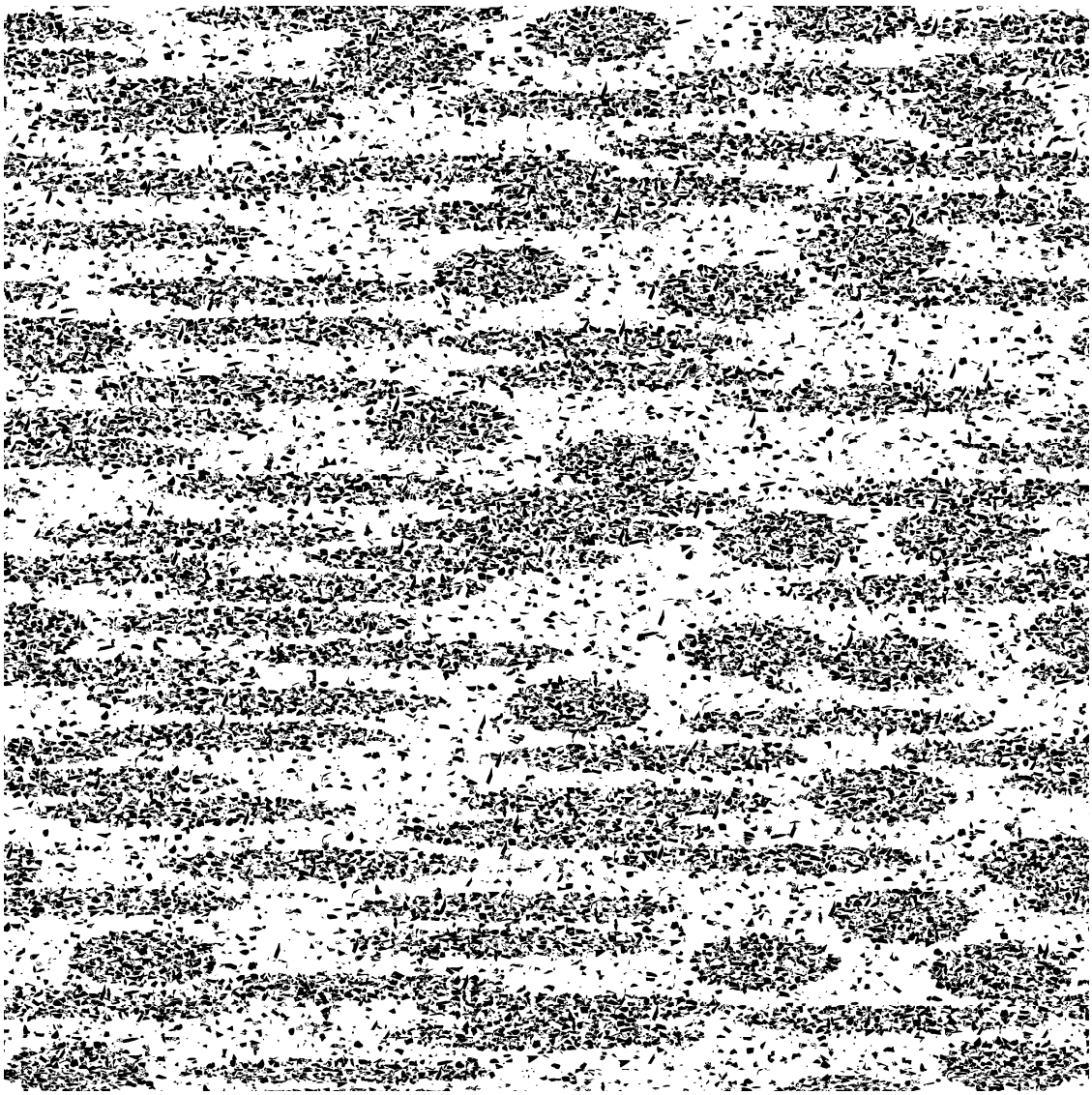


Figure 6.20: Simulated image for virtual microstructure with PSR = 8.0, volume fraction of SiC particles of 28% and having lower average SiC particle size than the corresponding real composites.

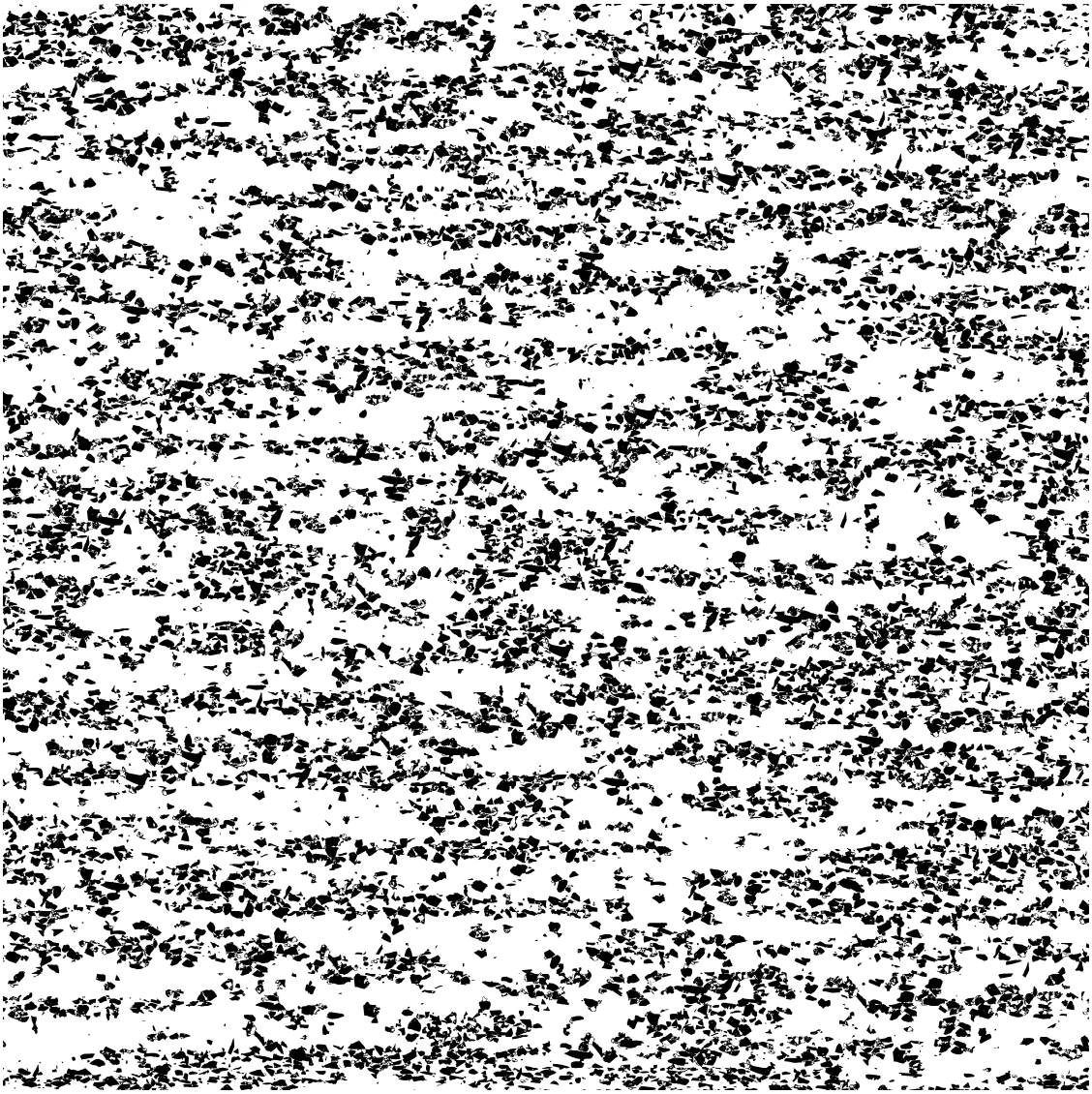


Figure 6.21a: Simulated image for virtual microstructure with PSR = 8.0, volume fraction of SiC particles of 28% and having higher extrusion ratio than the corresponding real microstructure.

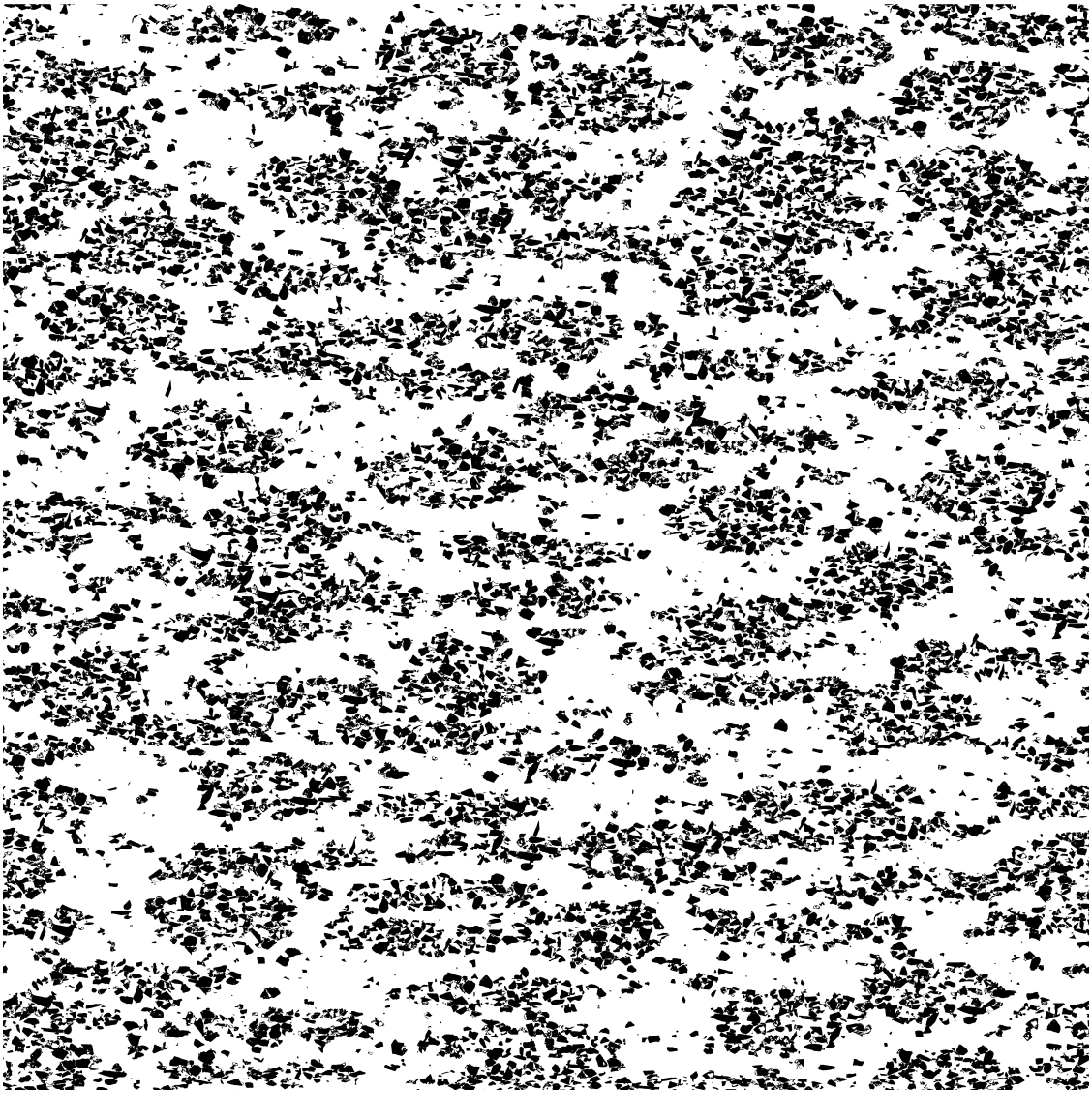


Figure 6.21b: Simulated image for virtual microstructure with PSR = 8.0, volume fraction of SiC particles of 28% and having lower extrusion ratio than the corresponding real microstructure.

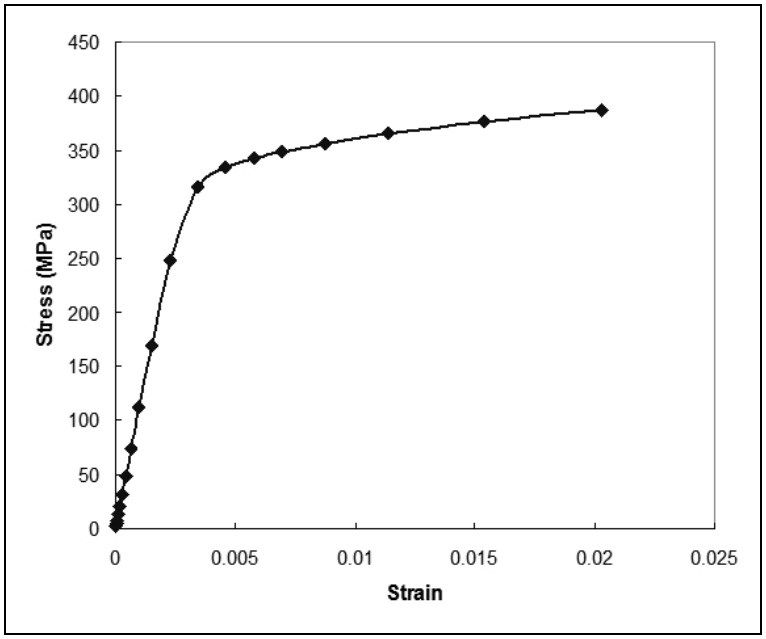



Figure 6.22: Stress-strain curve for virtual microstructure with PSR 6.0.

### 6.3 Simulations of "Realistic" Microstructures of Boron Modified Ti-Alloys

As mentioned earlier, the change of orientation of the SiC particles was not employed for the simulation of the DRA composites since the morphology of the particles is isotropic. Nevertheless, the simulation code is capable of allowing controlled rotation of the particles/whiskers and this feature of the code is utilized to simulate the "realistic" microstructures of boron modified Ti-alloys containing TiB whiskers. Firstly, realistic microstructures of compacted and extruded boron modified Ti alloys have been created, then, nine different extruded specimens, representing different extrusion parameters as shown in Table 6.2, have been utilized to employ the particle rotation model for creating realistic and virtual microstructures with varying extrusion parameters.

In the compacted alloy, the TiB whiskers have isotropic orientations, as shown in Figure 6.23a and the extruded alloy has TiB whiskers oriented in the direction of extrusion, as depicted in Figure 6.24a. Same simulation algorithm has been used to create "realistic" simulations for the microstructures of these alloys as for those of the DRA composites. The inputs for the code were a box of original TiB whiskers representing the size-shape distribution present in the real microstructures and the volume fraction of the TiB whiskers in the corresponding real microstructures. Figure 6.23b shows the simulated microstructure for the compacted boron modified Ti alloy and Figure 6.24b shows the simulated microstructure for the extruded boron modified Ti alloy. As can been seen from these images the simulated microstructures are qualitatively similar to those of corresponding real microstructure. The quantitative comparison between the real and simulated microstructures has been performed by using the two-point distribution functions and lineal path distribution functions. Figure 6.25 and 6.26 show the comparison of two-point probability functions between real and simulated microstructure of compacted and extruded boron modified Ti alloys, respectively and Figures 6.27 and 6.28 show the lineal path distribution comparison for the compacted and extruded Ti-alloys, respectively. As can be seen from these figures, the curves show a good match, implying that the simulated microstructures are statistically similar to the corresponding real microstructures.

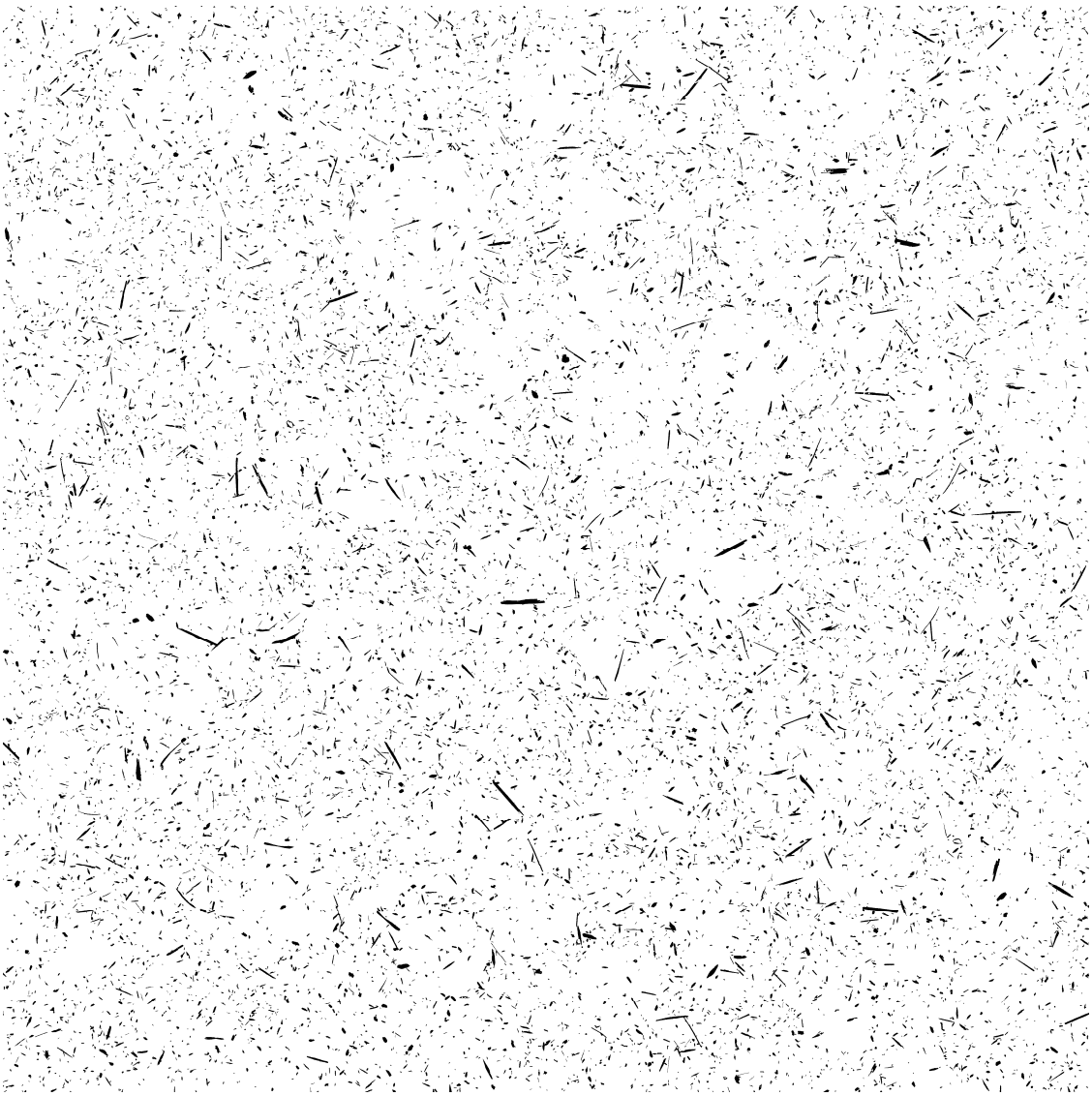


Figure 6.23: (a) Micrograph showing real microstructure of compacted boron modified Ti alloy.

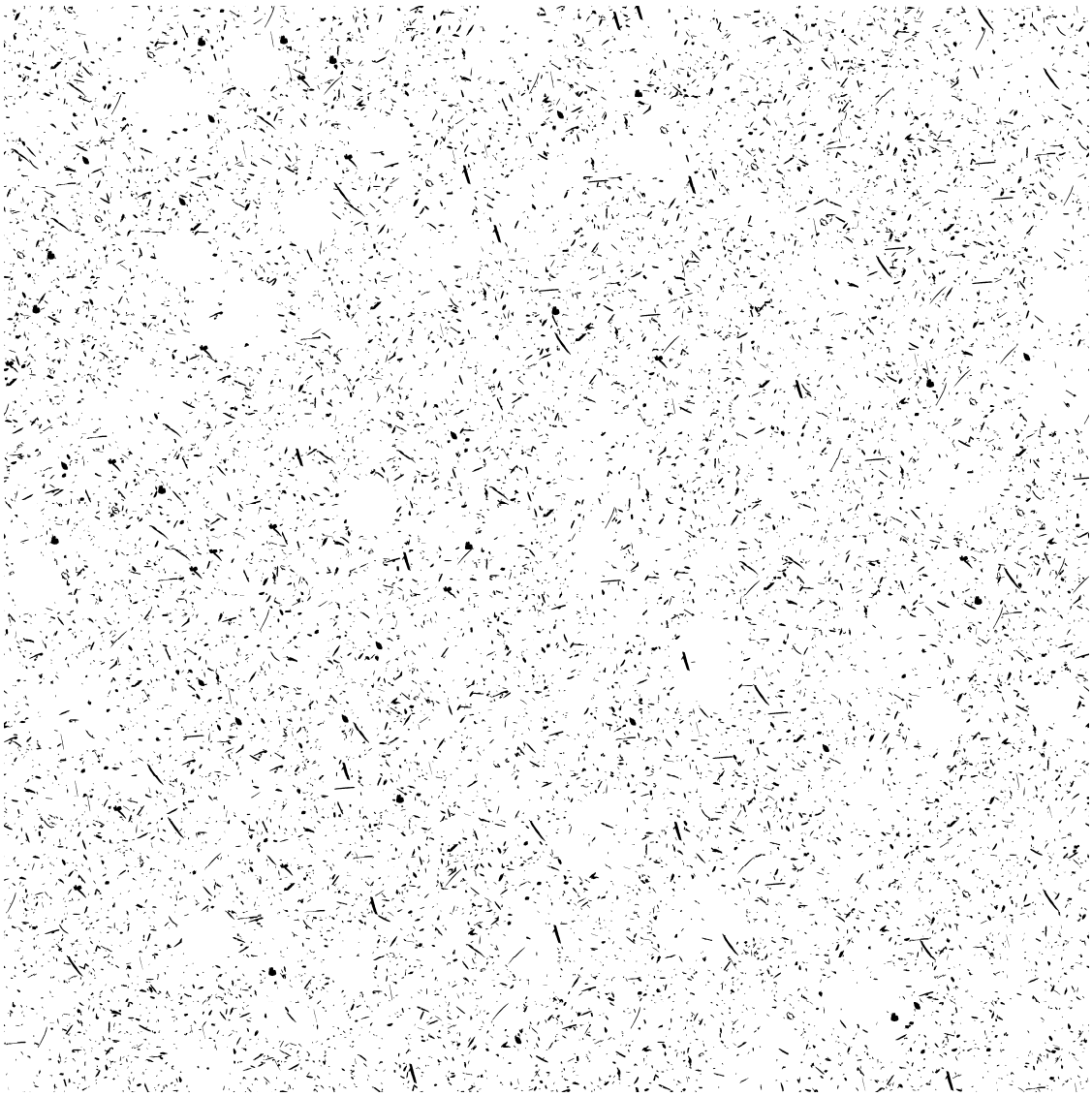


Figure 6.23: (b) Micrograph showing simulated microstructure of compacted boron modified Ti alloy.

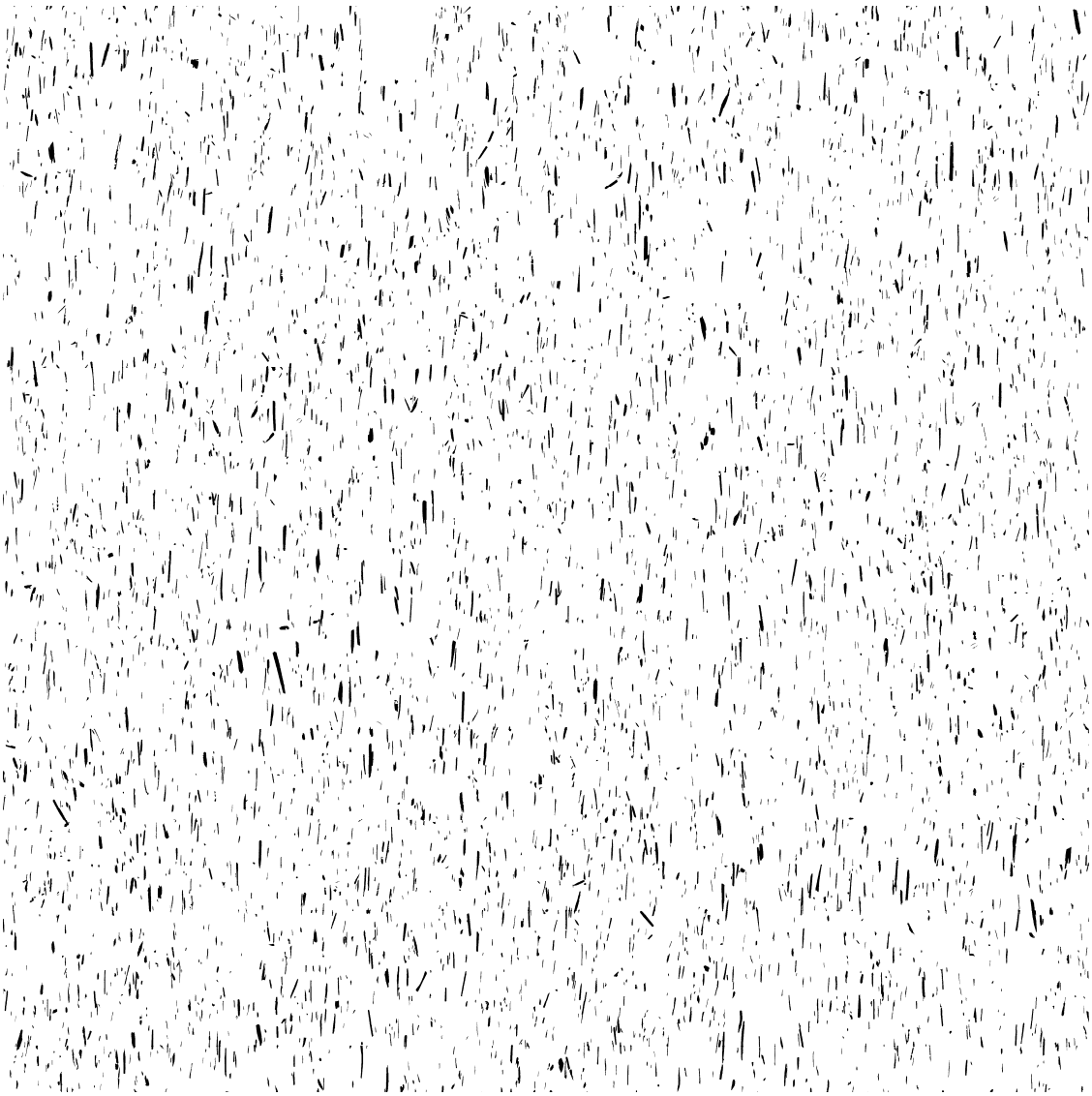


Figure 6.24: (a) Micrograph showing real microstructure of extruded boron modified Ti alloy.

.

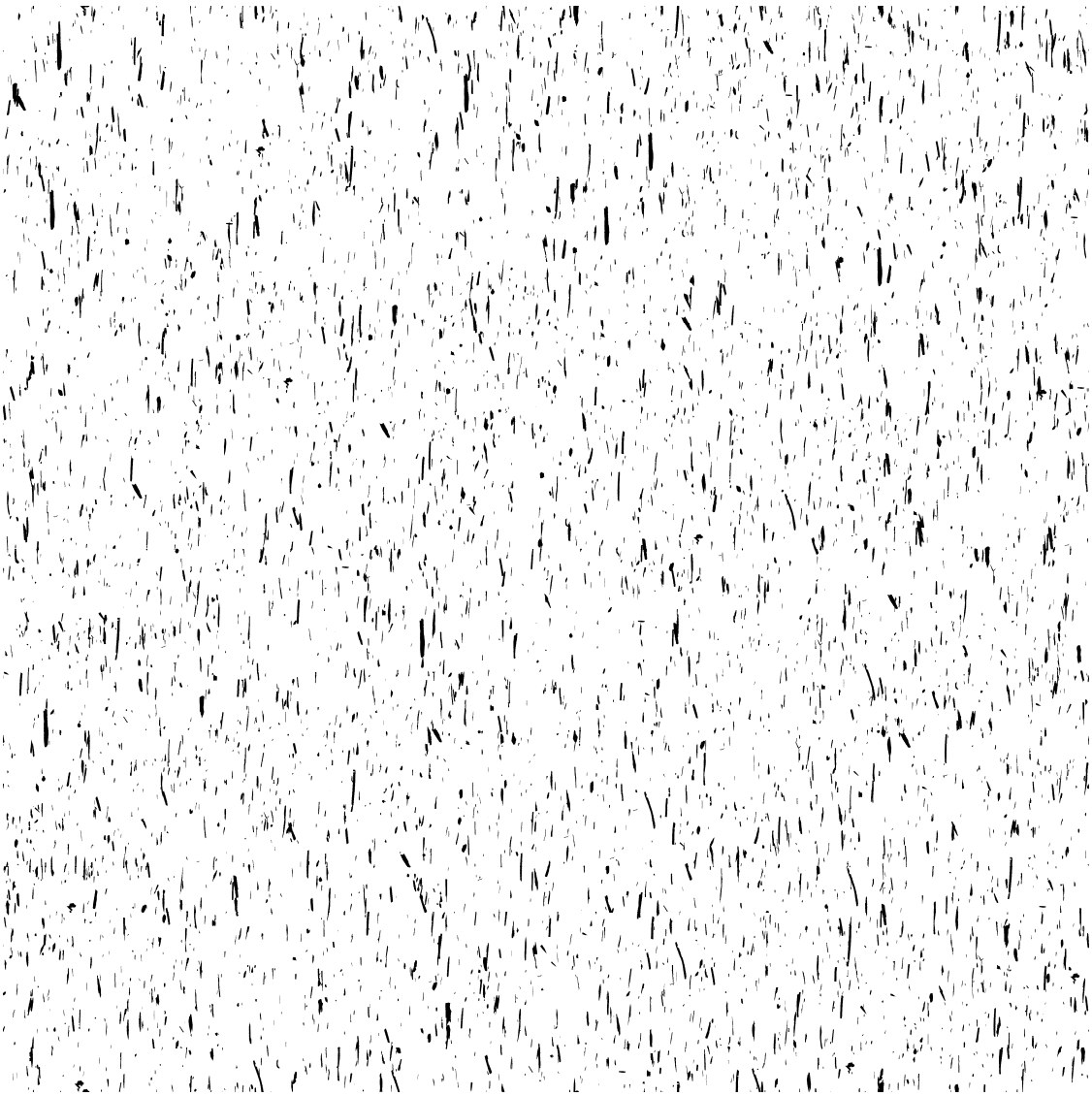


Figure 6.24: (b) Micrograph showing simulated microstructure of extruded boron modified Ti alloy.

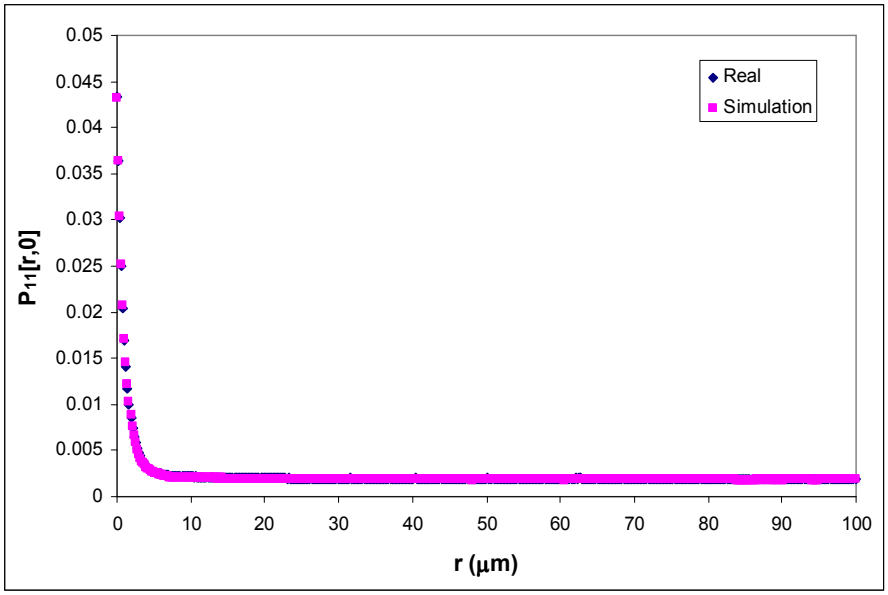



Figure 6.25a: Comparison of two point probability functions measured in the horizontal direction for real and simulated microstructures of compacted boron modified Ti alloy.

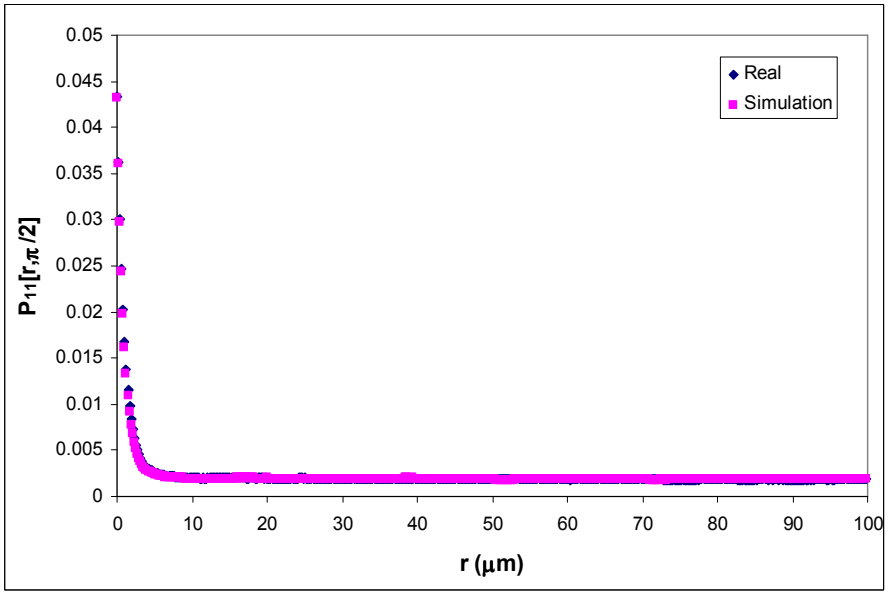



Figure 6.25b: Comparison of two point probability functions measured in the verticle direction for real and simulated microstructures of compacted boron modified Ti alloy.

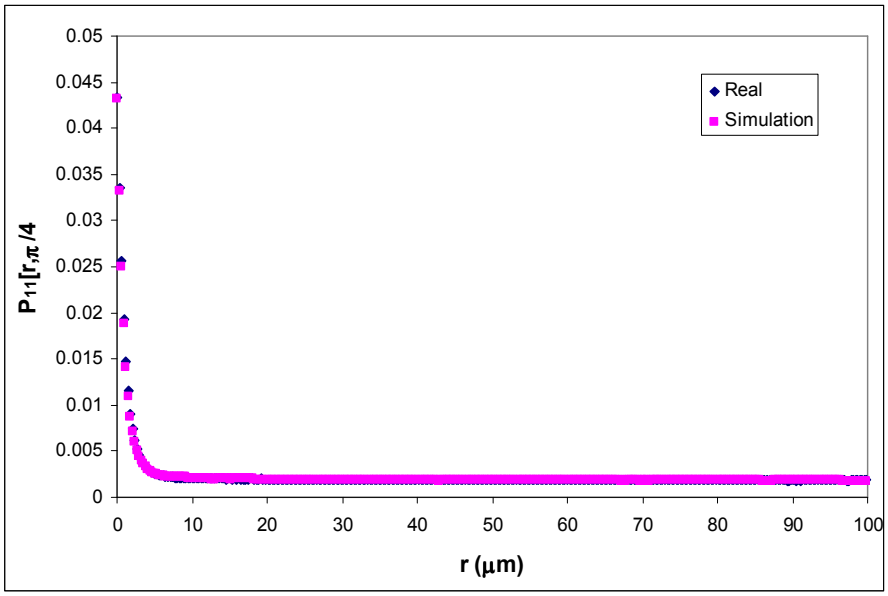



Figure 6.25c: Comparison of two point probability functions measured at an angle of 45° direction for real and simulated microstructures of compacted boron modified Ti alloy.

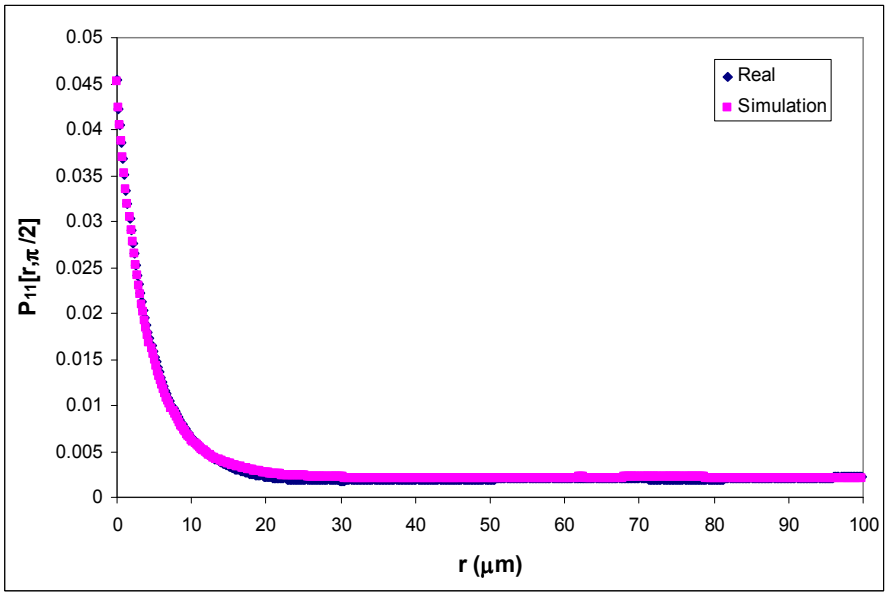



Figure 6.26a: Comparison of two point probability functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy.

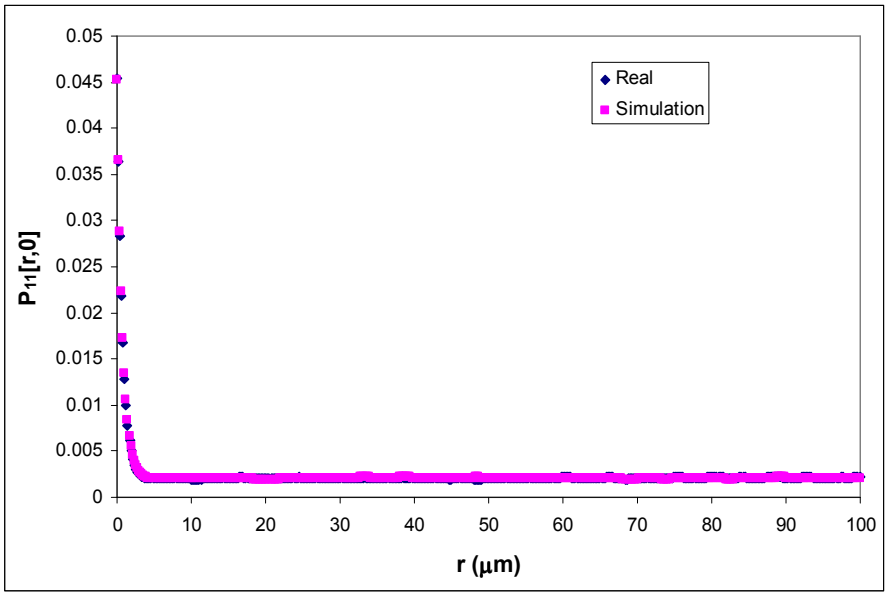



Figure 6.26b: Comparison of two point probability functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy.

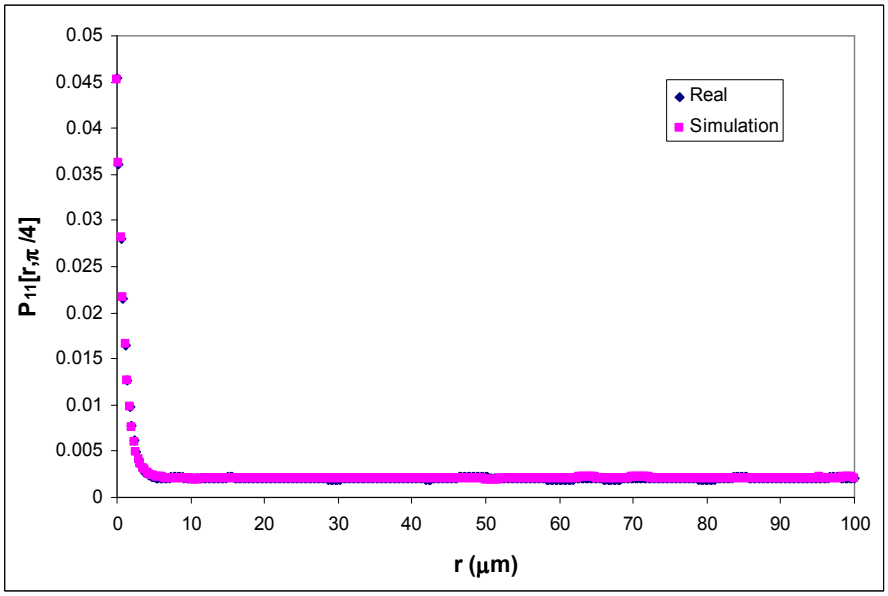



Figure 6.26c: Comparison of two point probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy

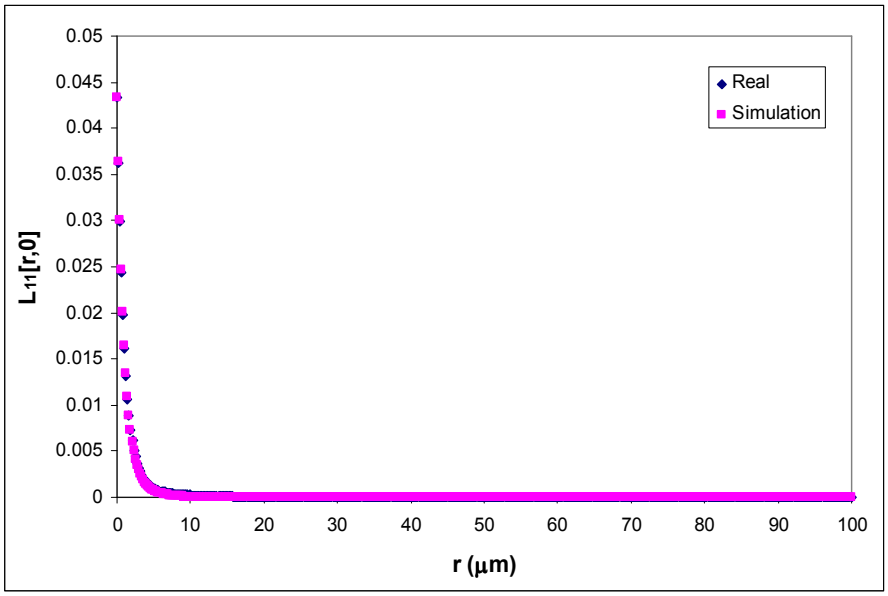



Figure 6.27a: Comparison of lineal path distribution functions measured in the horizontal direction for real and simulated microstructures of compacted boron modified Ti alloy.

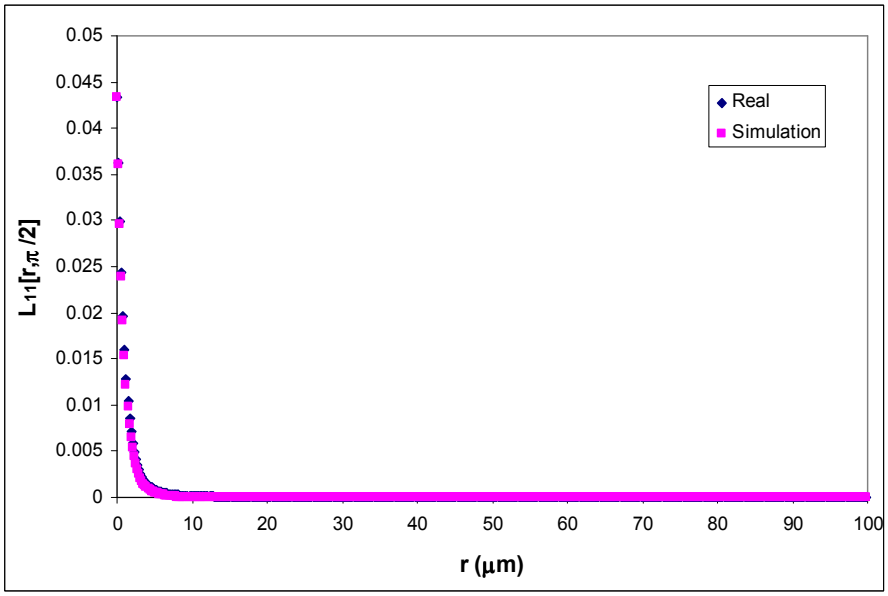



Figure 6.27b: Comparison of lineal path distribution functions measured in the vertical direction for real and simulated microstructures of compacted boron modified Ti alloy.

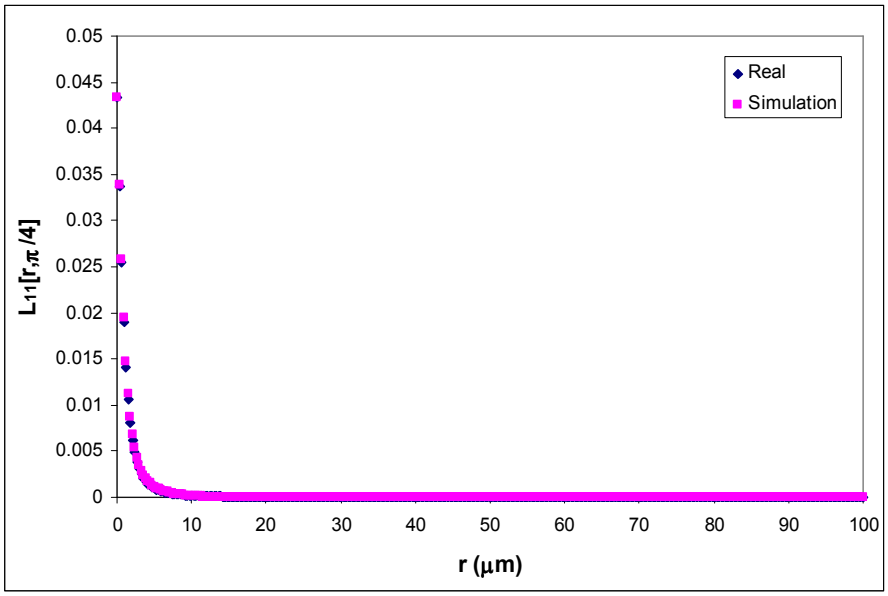



Figure 6.27c: Comparison of lineal path distribution functions measured at an angle of 45° direction for real and simulated microstructures of compacted boron modified Ti alloy.

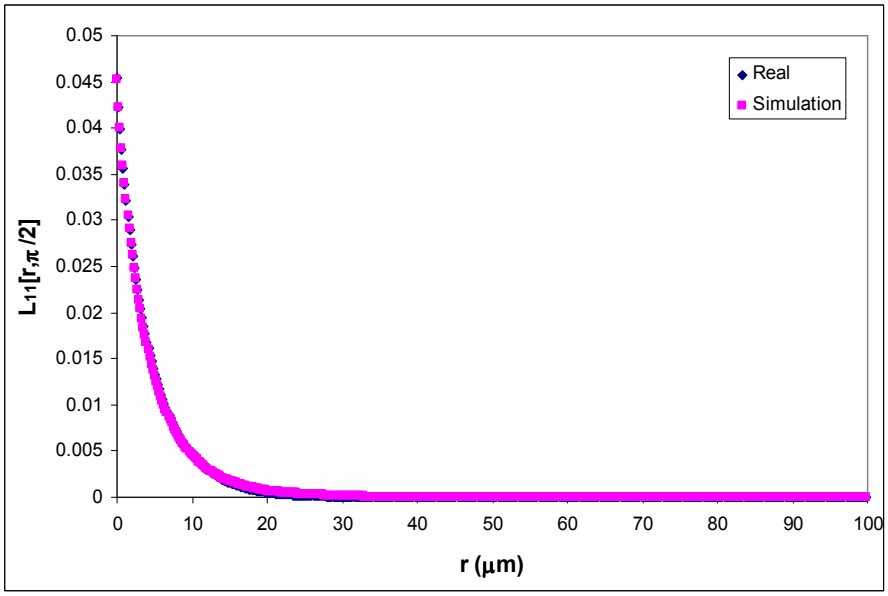



Figure 6.28a: Comparison of lineal path distribution functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy.

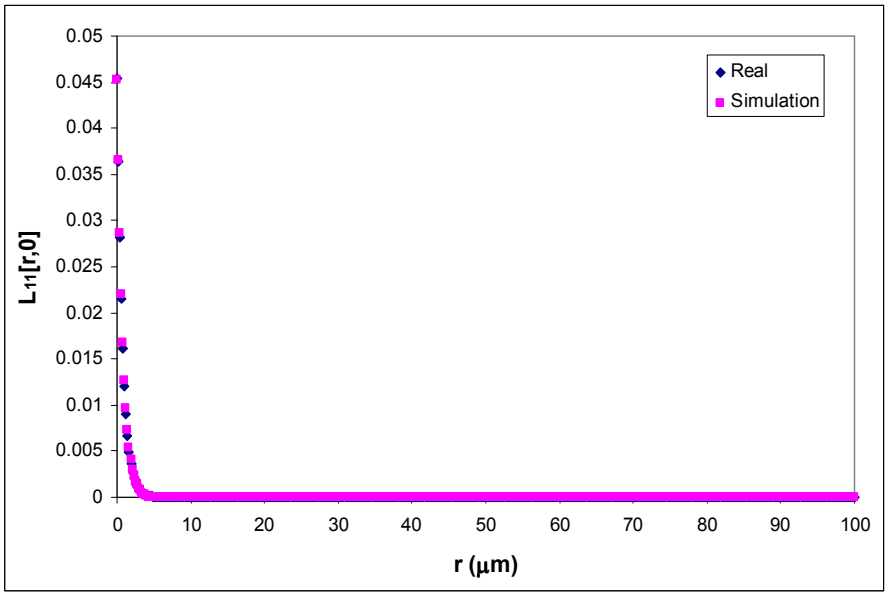



Figure 6.28b: Comparison of lineal path distribution functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy.

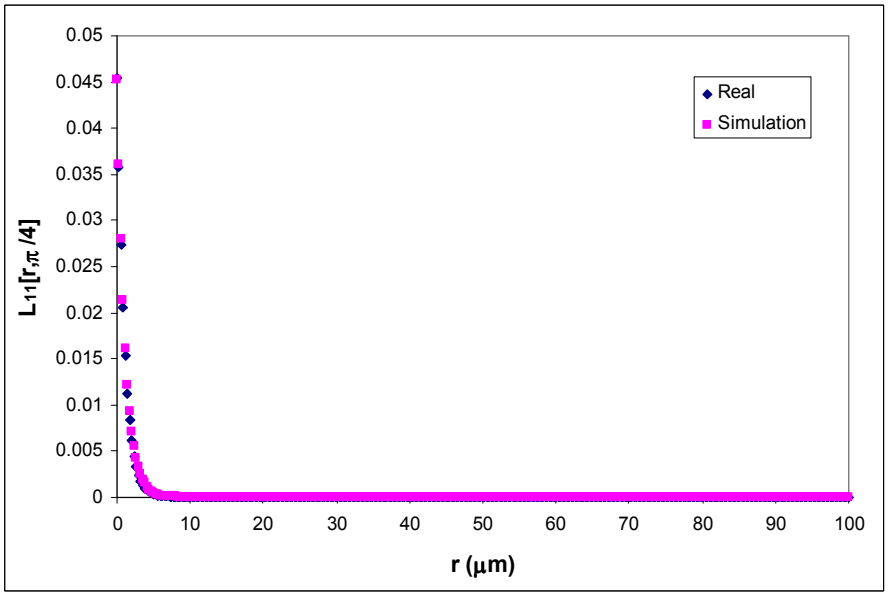



Figure 6.28c: Comparison of lineal path distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy.

Table 6.2: Extrusion parameters for extruded boron modified Ti-6Al-4V alloy

| Sample ID | Extrusion Temperature (°C) | Extrusion Ratio |
|---|---|---|
| A | 1010 | 8:1 |
| B | 1010 | 12:1 |
| C | 1010 | 16:1 |
| | | |
| D | 1057 | 8:1 |
| E | 1057 | 12:1 |
| F | 1057 | 16:1 |
| | | |
| G | 1104 | 8:1 |
| H | 1104 | 12:1 |
| I | 1104 | 16:1 |

As can be seen from the Figure 6.24, the TiB whiskers have high aspect ratio, which makes them amenable to rotation during processing such as extrusion. Furthermore, the extent of rotation of the TiB whiskers is dependent on the processing parameters such as extrusion temperature and extrusion ratio. In the present work, nine different specimens with varying extrusion parameters, as shown in Table 6.2, have been used to establish relationships between extrusion parameters and simulation parameters. These relationships can then utilized to create virtual microstructures of boron modified Ti-alloys with extrusion conditions different than the given real microstructures. Firstly, realistic microstructures for all the corresponding real microstructures have been simulated using the simulation code described in the previous Chapter. The input box of particles was obtained from the sample A and it contained TiB whiskers covering the size-shape distribution of TiB whiskers in all the specimens. Since, in addition to the change in the level of clustering, the change in extrusion parameters also cause the TiB whiskers to orient with varying degrees, a simulation parameter of ‘rotation angle’ have been added to the simulation model. This parameter defines the amount of rotation to be

applied to the TiB whiskers during simulation. Figures 6.29a to 6.37a show the real microstructures of the specimens A to I, respectively, and Figures 6.29b to 6.37b show the corresponding simulated microstructures. The qualitative comparison of the corresponding real and simulated microstructure reveals their visual similarity and the quantitative comparisons have been performed using the two-point probability and lineal path distribution functions. Figures 6.38 to 6.46 show the comparison of the two-point probability functions and Figures 6.47 to 6.55 show the comparison of lineal path distribution functions for the corresponding real and simulated microstructures. As can be seen from these comparisons, the functions show a good match for the real and simulated microstructures, implying the statistical similarity between the corresponding microstructures. Table 6.3 shows the list of simulation parameters used to simulate realistic microstructures for the extruded Ti-alloy specimens. As can be noted from the Table 6.3, for a given temperature, the value for 'rotation angle' increases as the extrusion ratio is increased. This relationship confirms the physical behavior of increased alignment of the TiB whiskers in the extrusion direction as the extrusion ratio is increased. Furthermore, the length of the clusters increases as the extrusion ratio is increased, which also depicts the physical nature of the extrusion process. The clustering intensity decreases as the extrusion temperature is increased, revealing the more uniform distribution of the TiB whiskers for extrusions at higher temperatures. These parameters can now be utilized to generate virtual microstructures of extruded boron modified Ti-alloys with a variety extrusion parameters other than the given set for the real microstructures. Figure 5.56 shows the relationship between extrusion ratio and cluster length and Figure 5.57 shows the relationship between extrusion ratio and rotation angle,

for the extrusion temperature 1010 °C. These relationships have been utilized to generate a virtual microstructure with extrusion temperature of 1010 °C and extrusion ratio of 14:1. The simulation parameters for this virtual microstructure have been calculated from the simulation parameters used for the simulation of realistic microstructures of Ti-alloys with extrusion temperature 1010 °C. The length of the clusters has been altered to 440 μm and the rotation angle of 6.8 degrees has been used to generate the virtual microstructure shown in Figure 6.58. Figure 6.59 shows the relationship between extrusion temperature and clustering intensity. This relationship has been utilized in generating another virtual microstructure with extrusion temperature 1080 °C and extrusion ratio of 8:1 have been created, as shown in Figure 6.60. The clustering intensity of 3.5 has been used to create this virtual microstructure. These virtual microstructures can be utilized to study the mechanical behavior of these alloys representing a wide range of processing parameters without physical manufacturing the specimens.

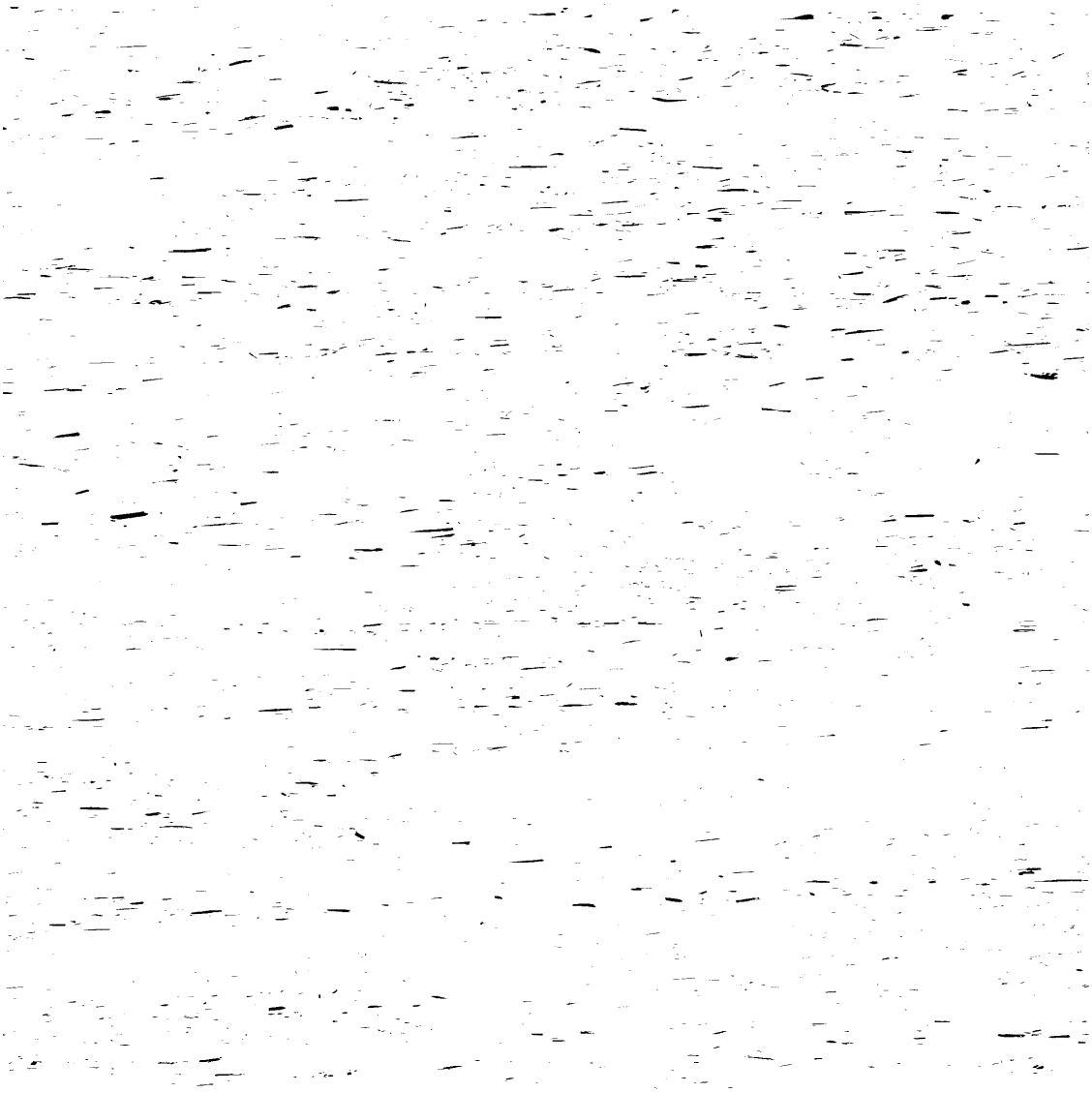


Figure 6.29: (a) Micrograph showing real microstructure of extruded boron modified Ti alloy specimen A.

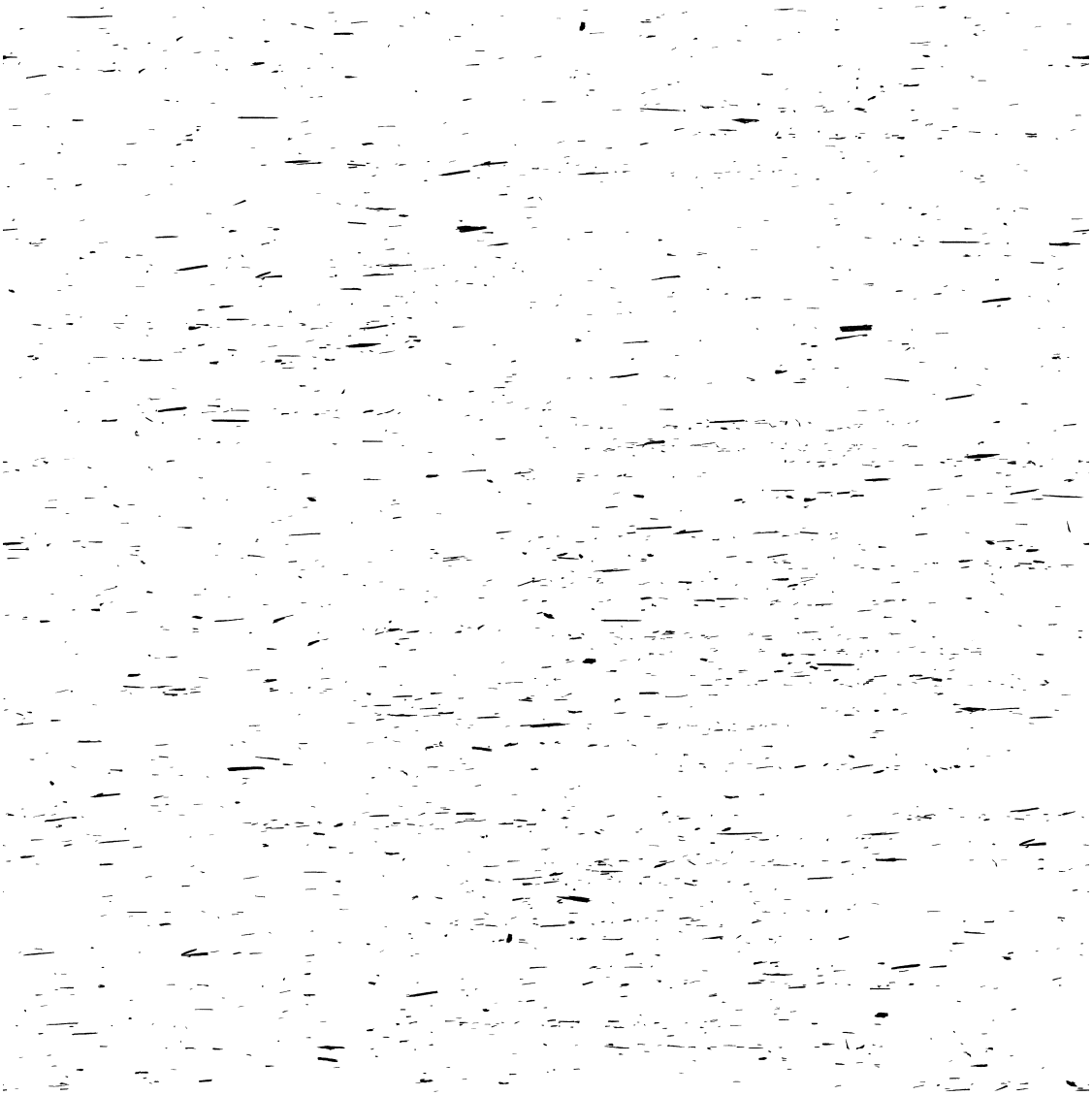


Figure 6.29: (b) Micrograph showing simulated microstructure of extruded boron modified Ti alloy specimen A.

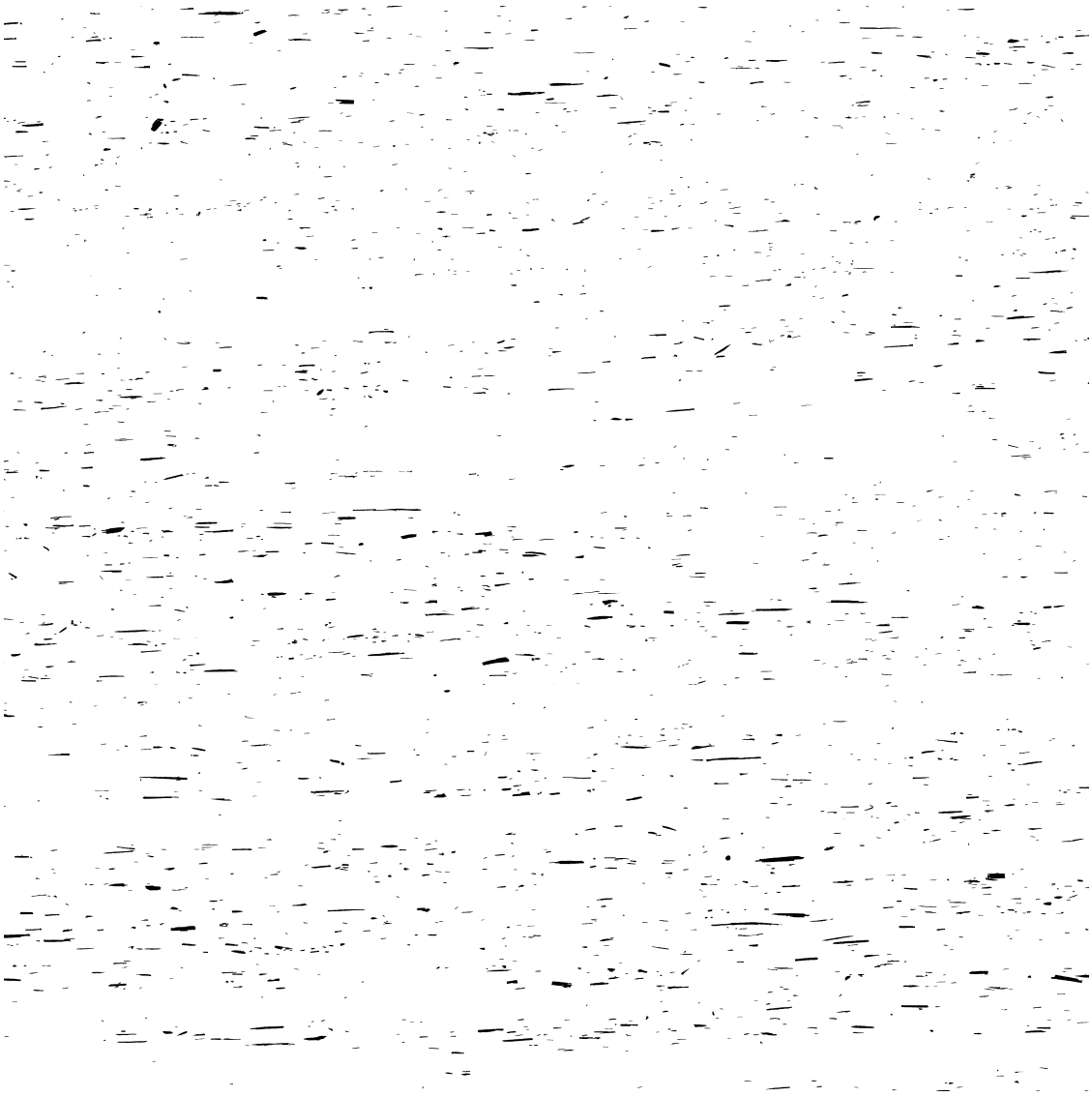


Figure 6.30: (a) Micrograph showing real microstructure of extruded boron modified Ti alloy specimen B.

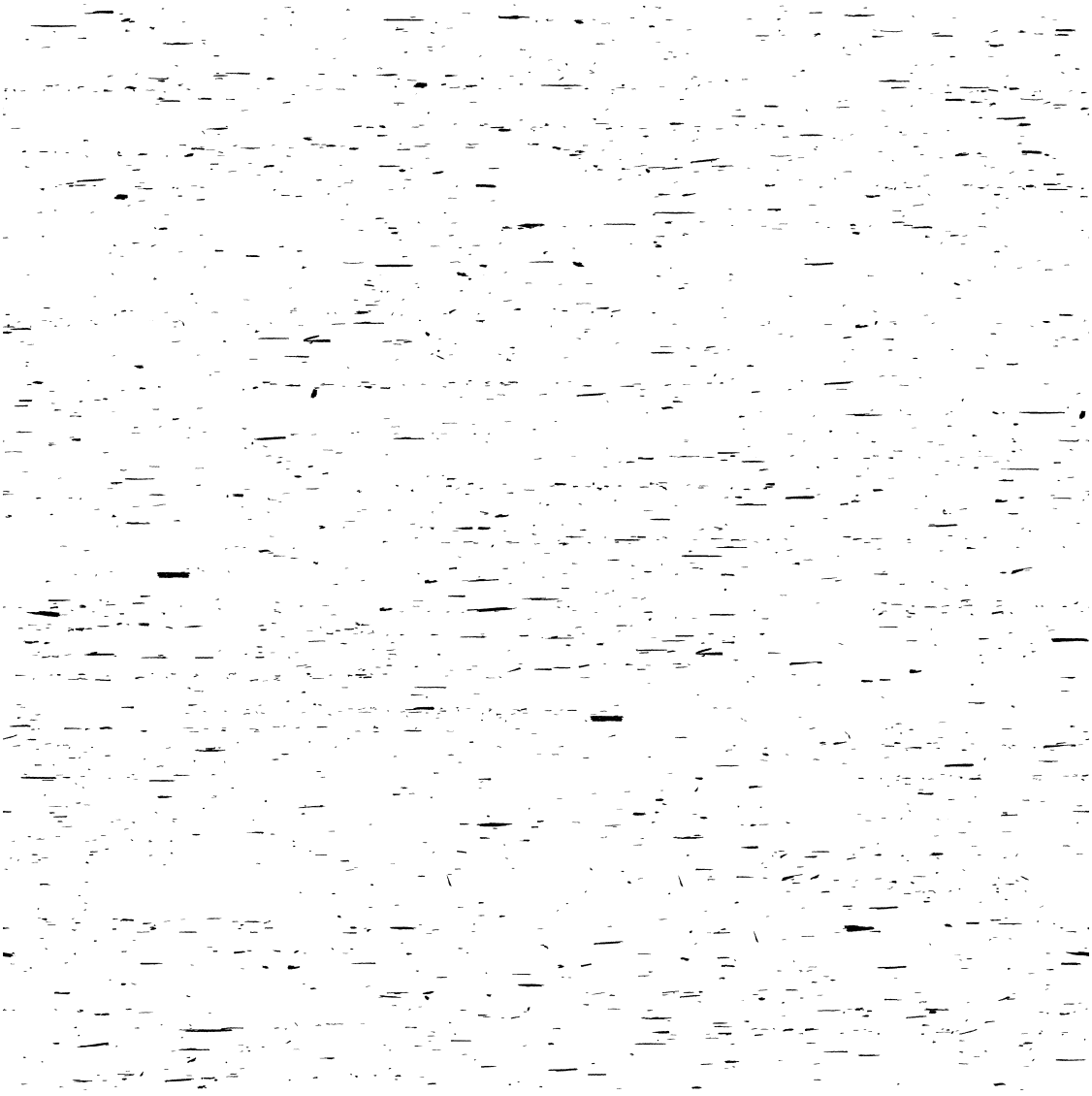


Figure 6.30: (b) Micrograph showing simulated microstructure of extruded boron modified Ti alloy specimen B.

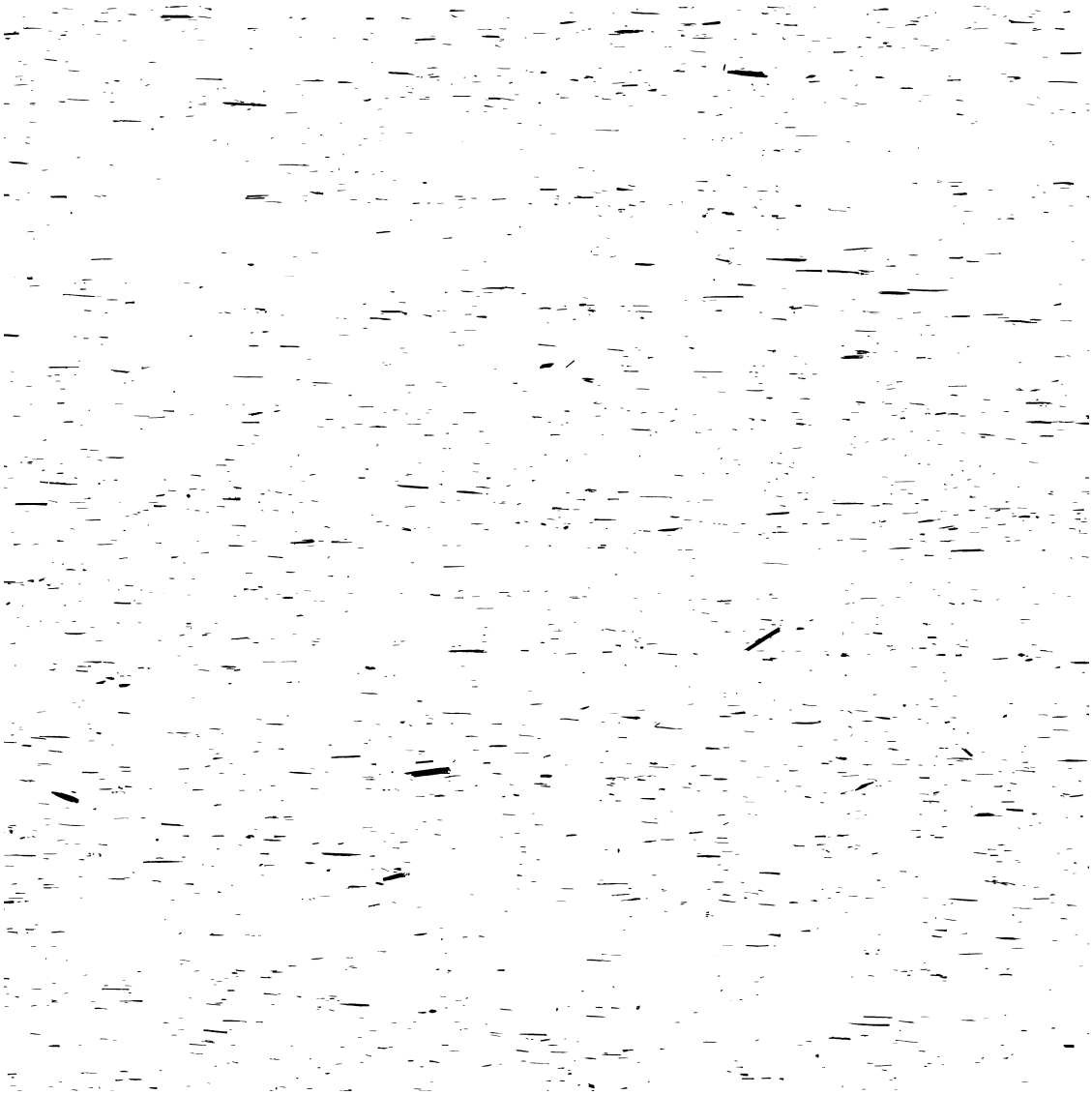


Figure 6.31: (a) Micrograph showing real microstructure of extruded boron modified Ti alloy specimen C.

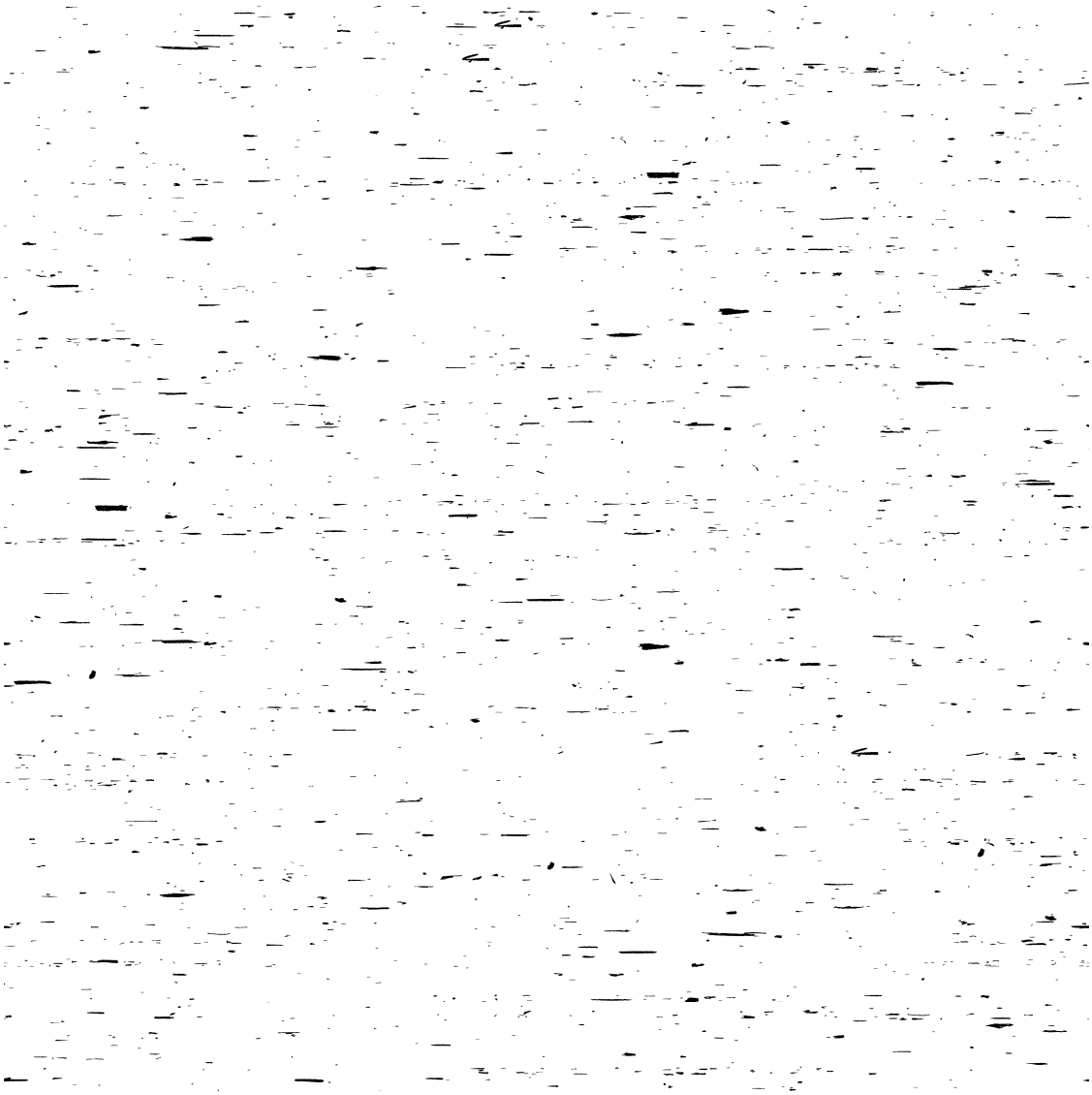


Figure 6.31: (b) Micrograph showing simulated microstructure of extruded boron modified Ti alloy specimen C.

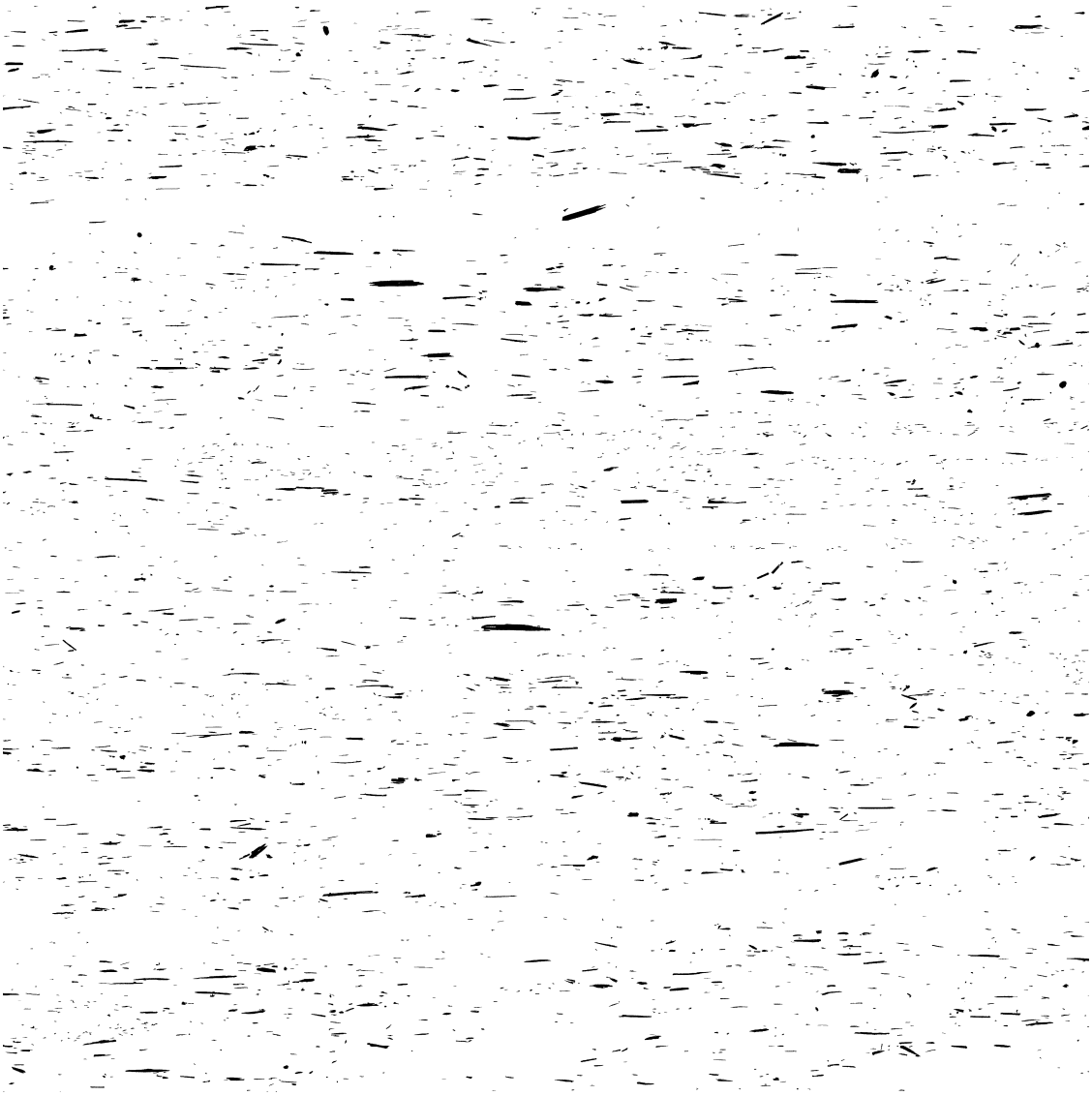


Figure 6.32: (a) Micrograph showing real microstructure of extruded boron modified Ti alloy specimen D.

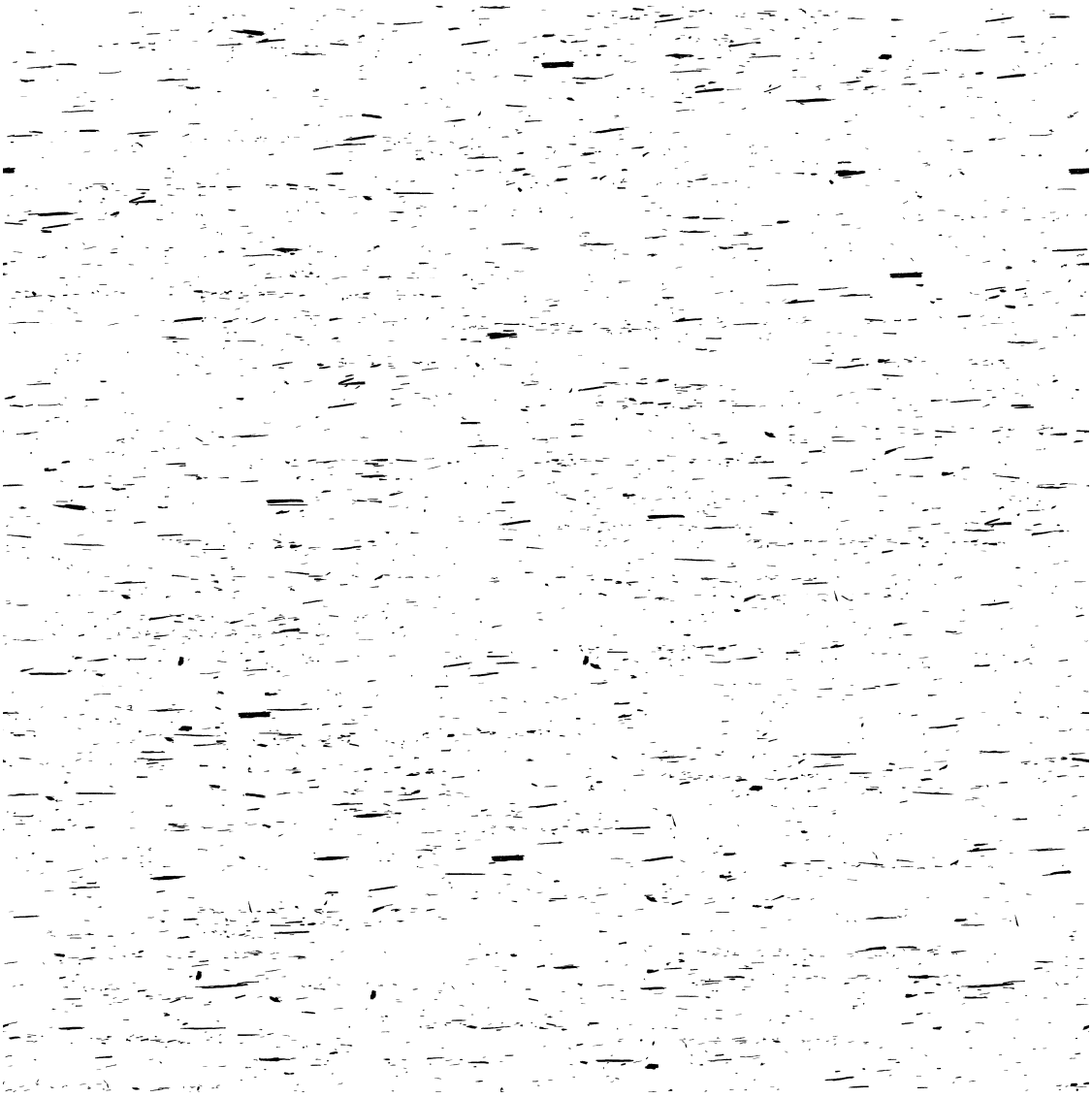


Figure 6.32: (b) Micrograph showing simulated microstructure of extruded boron modified Ti alloy specimen D.

Figure 6.33: (a) Micrograph showing real microstructure of extruded boron modified Ti alloy specimen E.

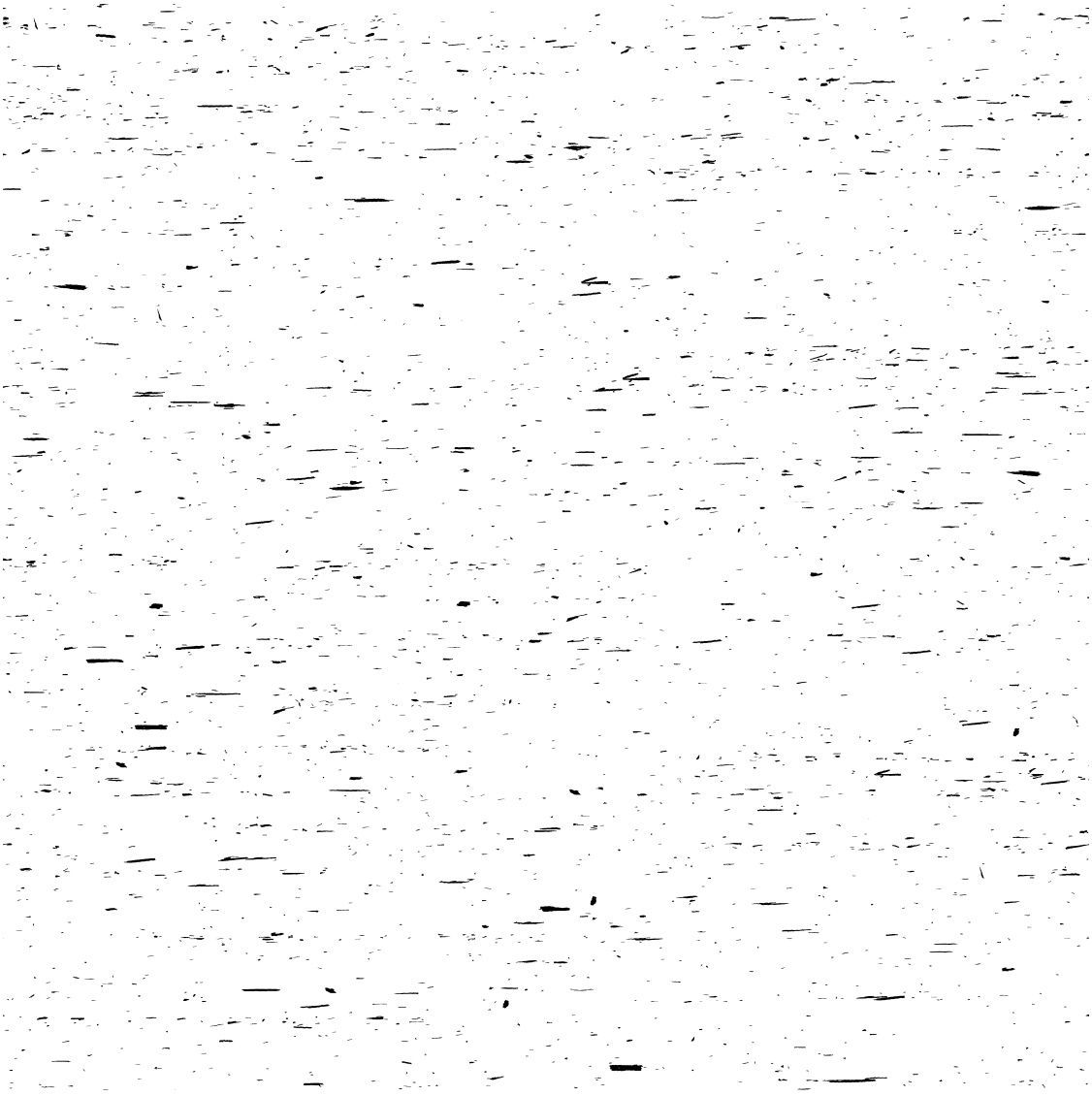


Figure 6.33: (b) Micrograph showing simulated microstructure of extruded boron modified Ti alloy specimen E.

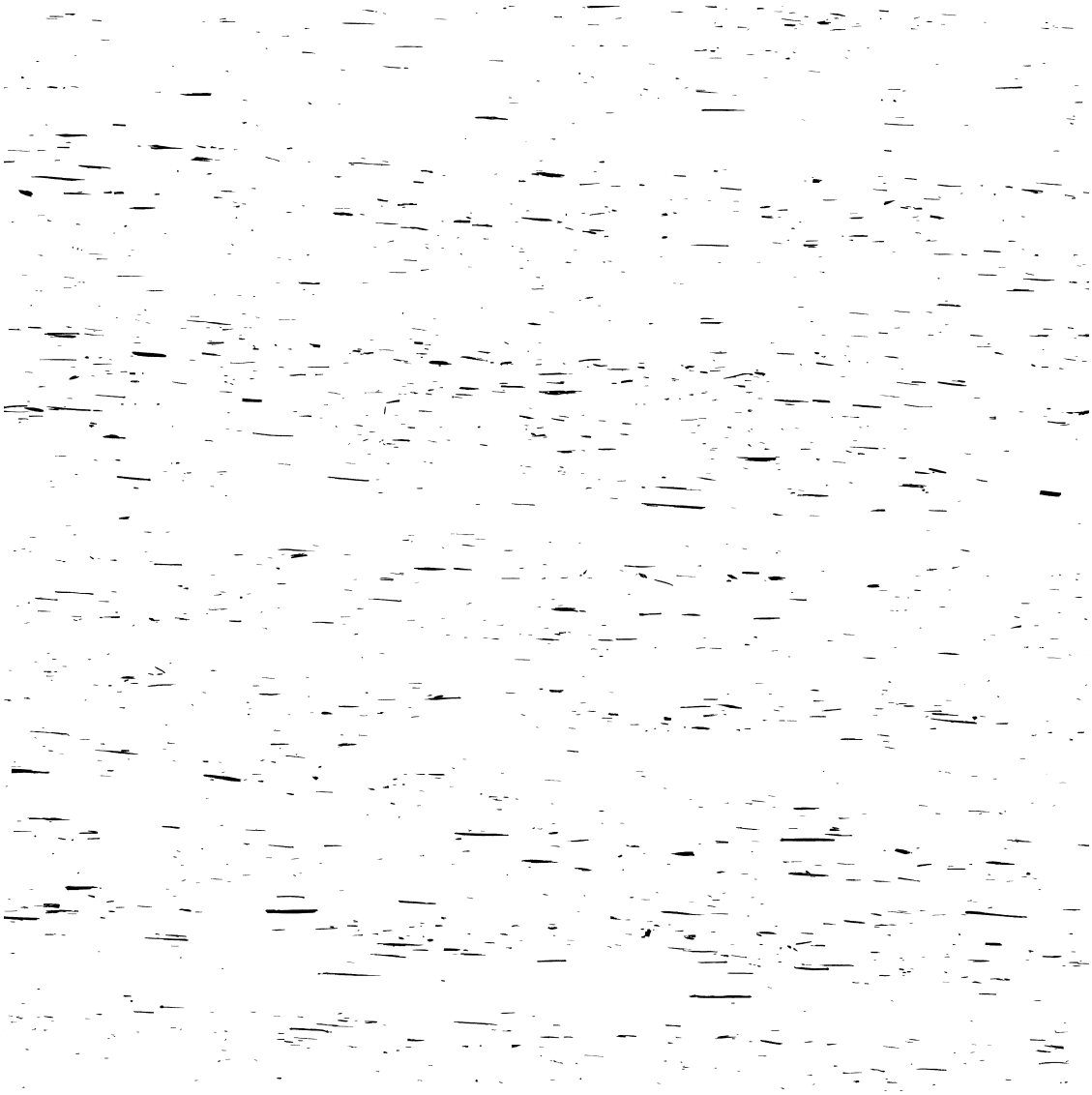


Figure 6.34: (a) Micrograph showing real microstructure of extruded boron modified Ti alloy specimen F.

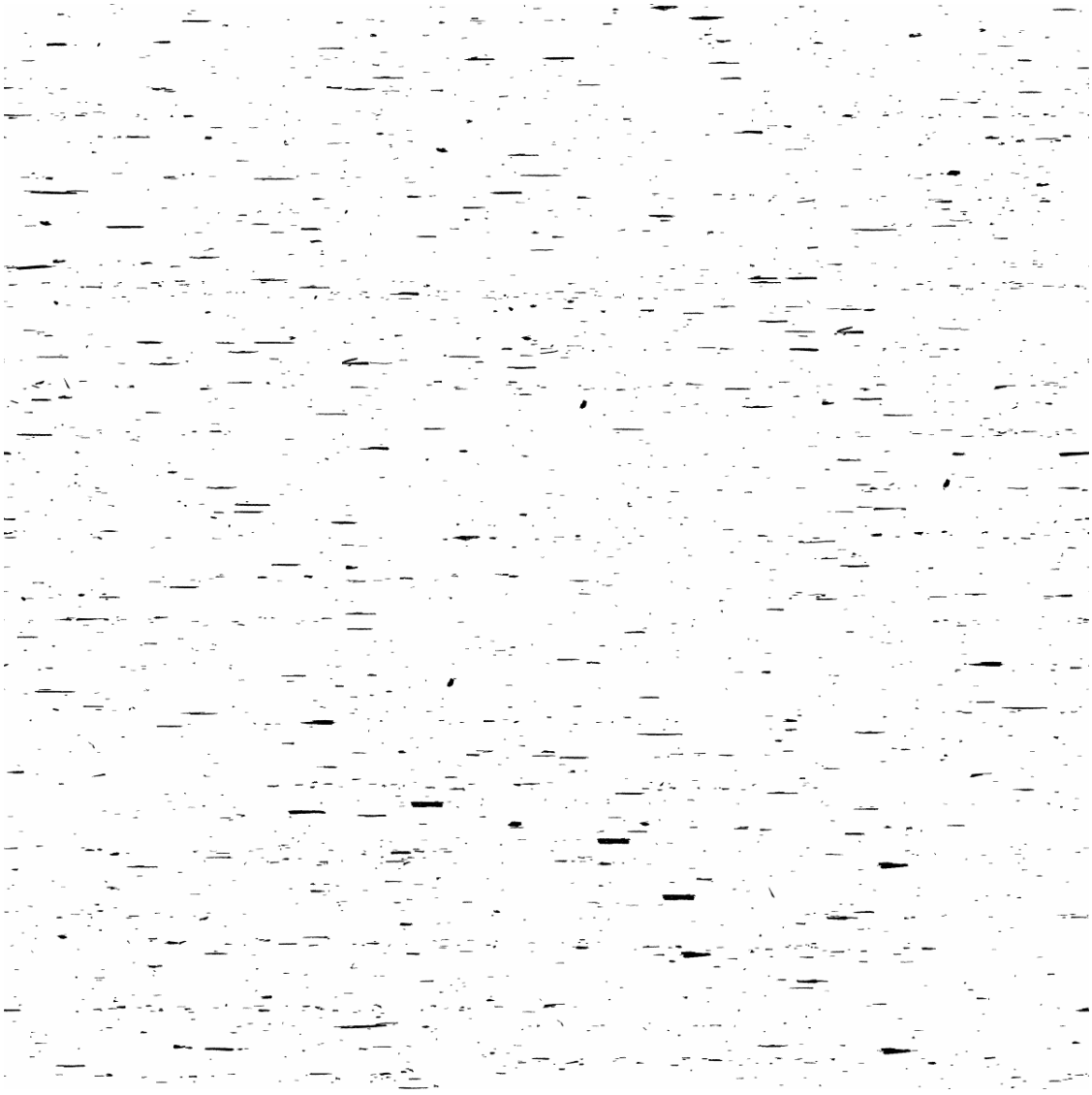


Figure 6.34: (b) Micrograph showing simulated microstructure of extruded boron modified Ti alloy specimen F.

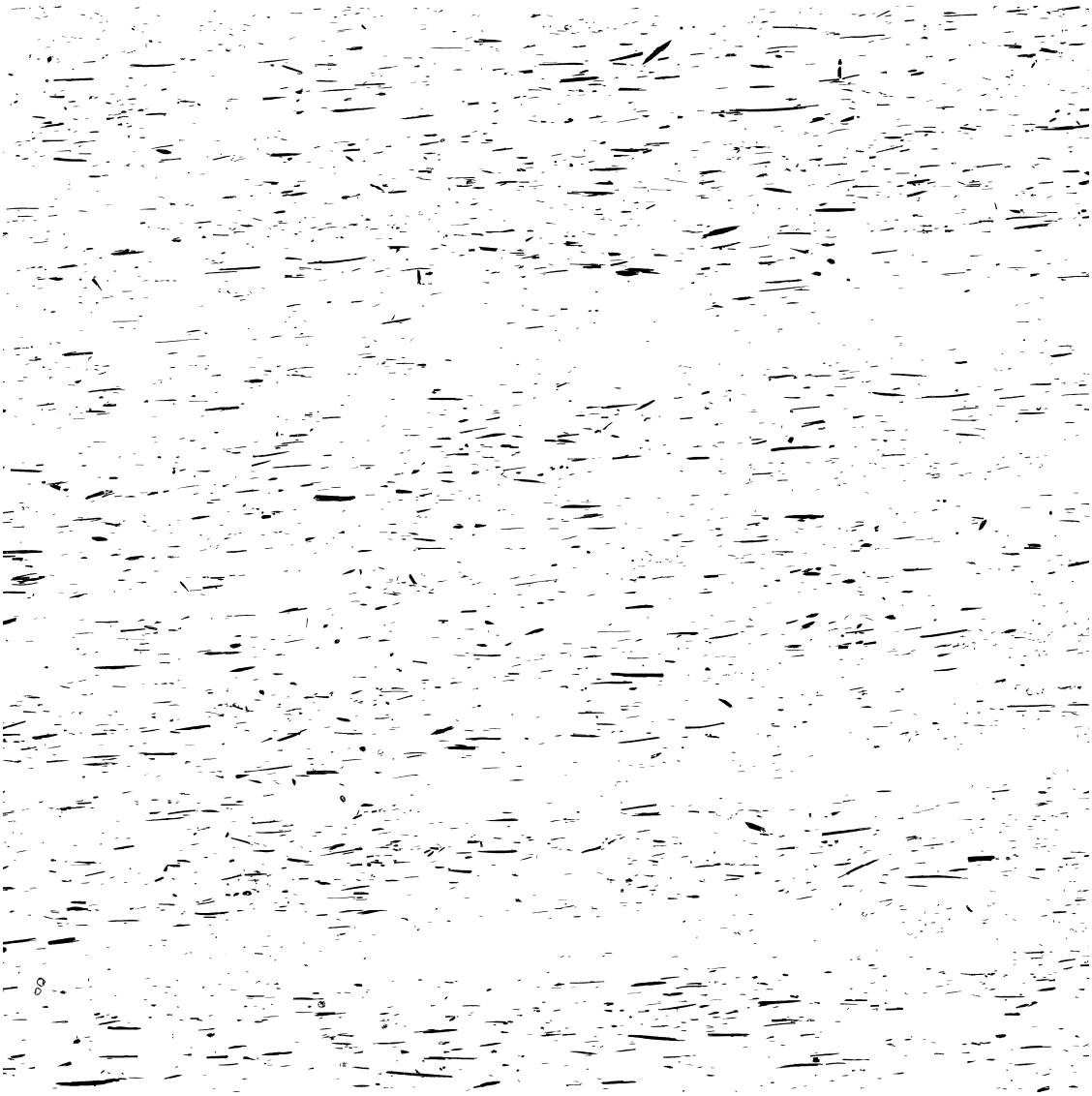


Figure 6.35: (a) Micrograph showing real microstructure of extruded boron modified Ti alloy specimen G.

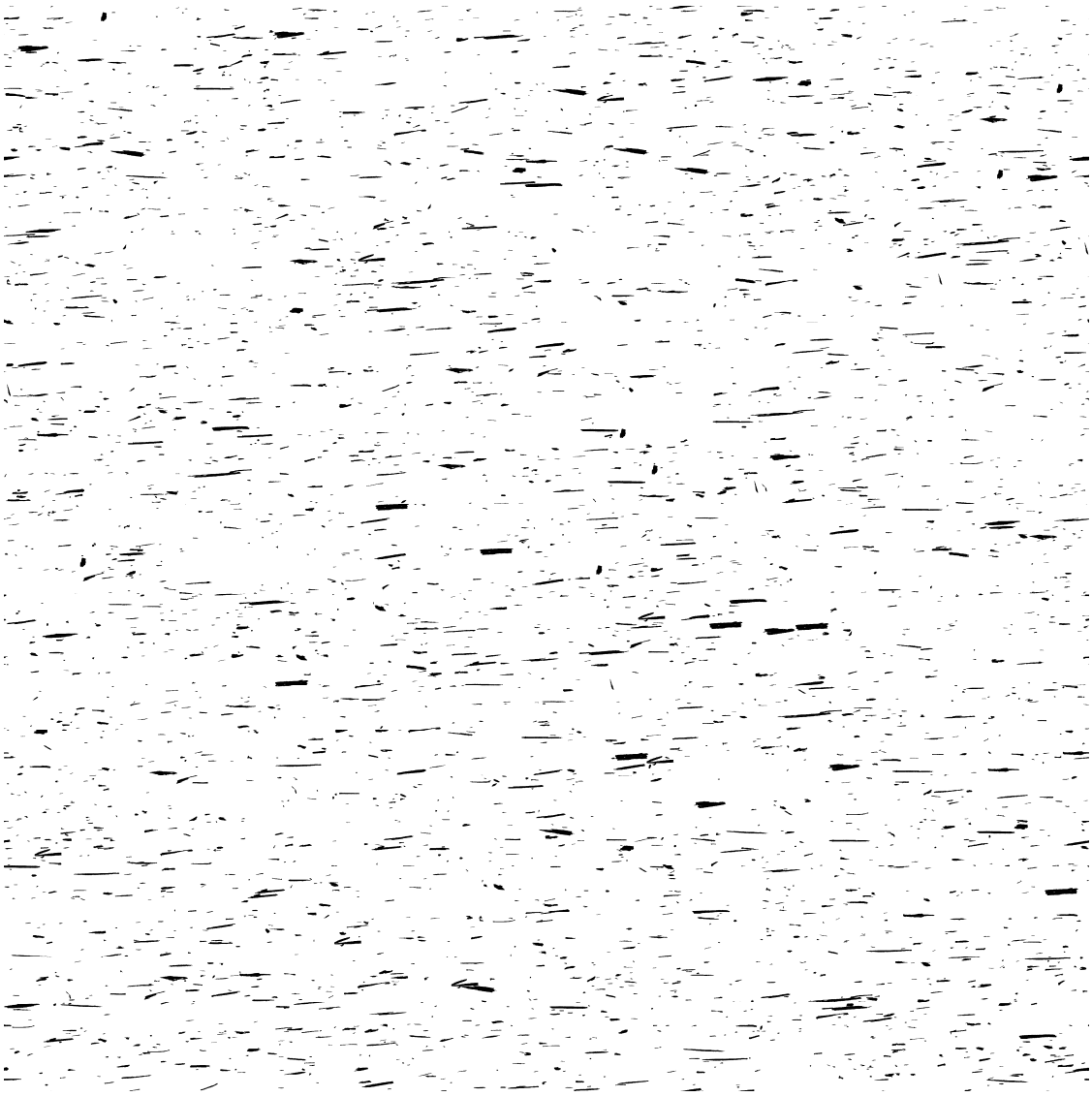


Figure 6.35: (b) Micrograph showing simulated microstructure of extruded boron modified Ti alloy specimen G.

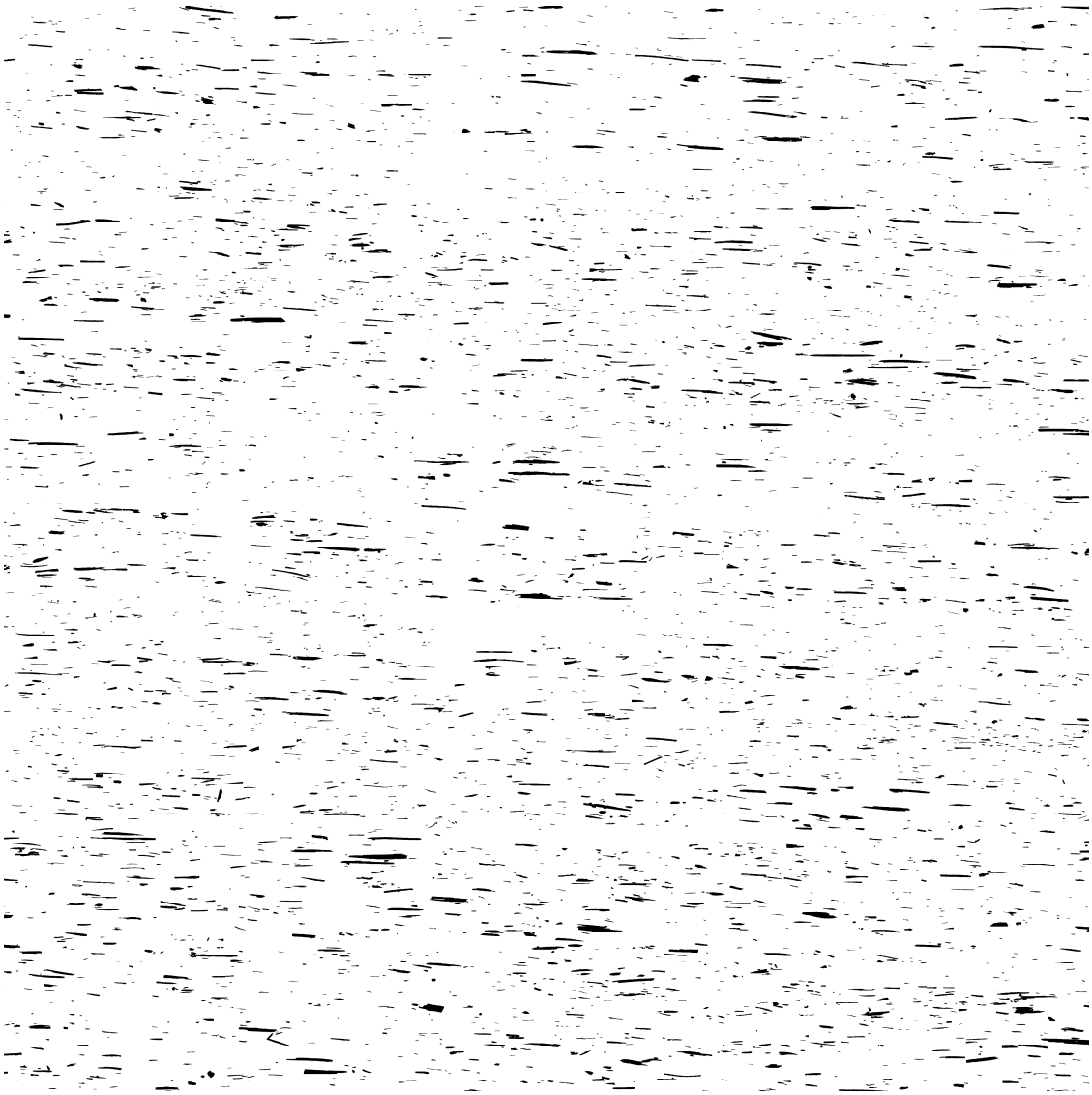


Figure 6.36: (a) Micrograph showing real microstructure of extruded boron modified Ti alloy specimen H.

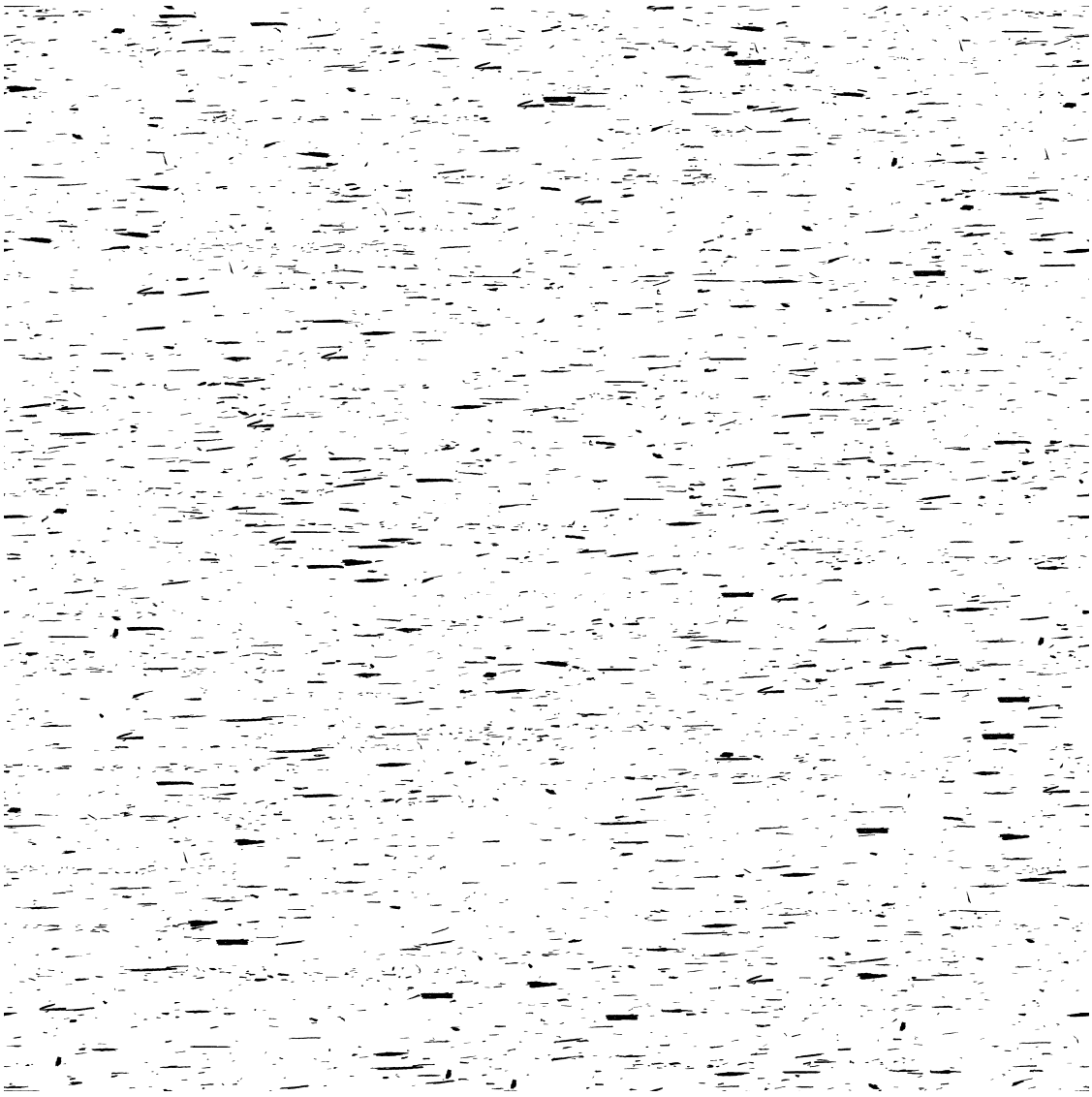


Figure 6.36: (b) Micrograph showing simulated microstructure of extruded boron modified Ti alloy specimen H.

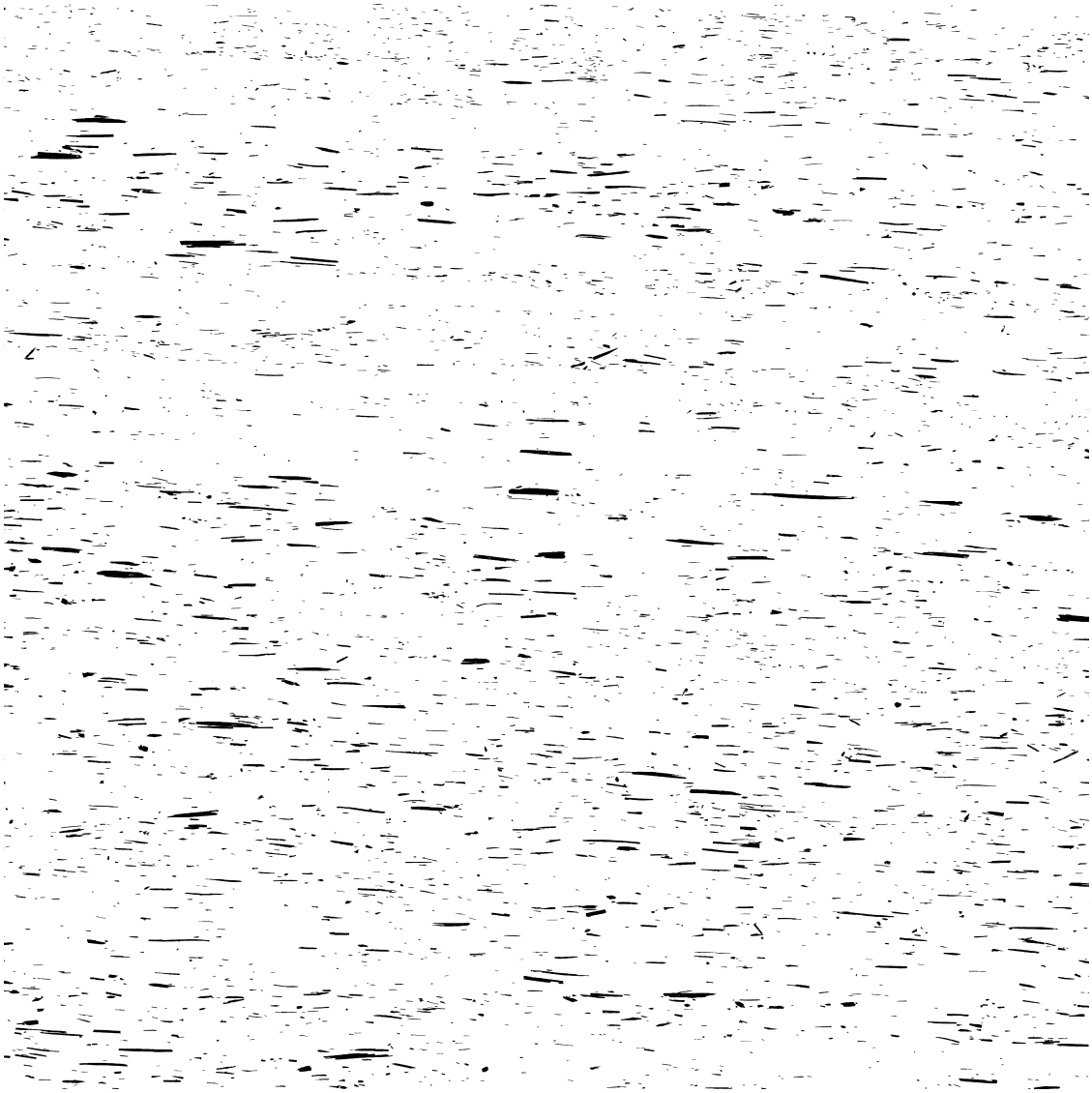


Figure 6.37: (a) Micrograph showing real microstructure of extruded boron modified Ti alloy specimen I.

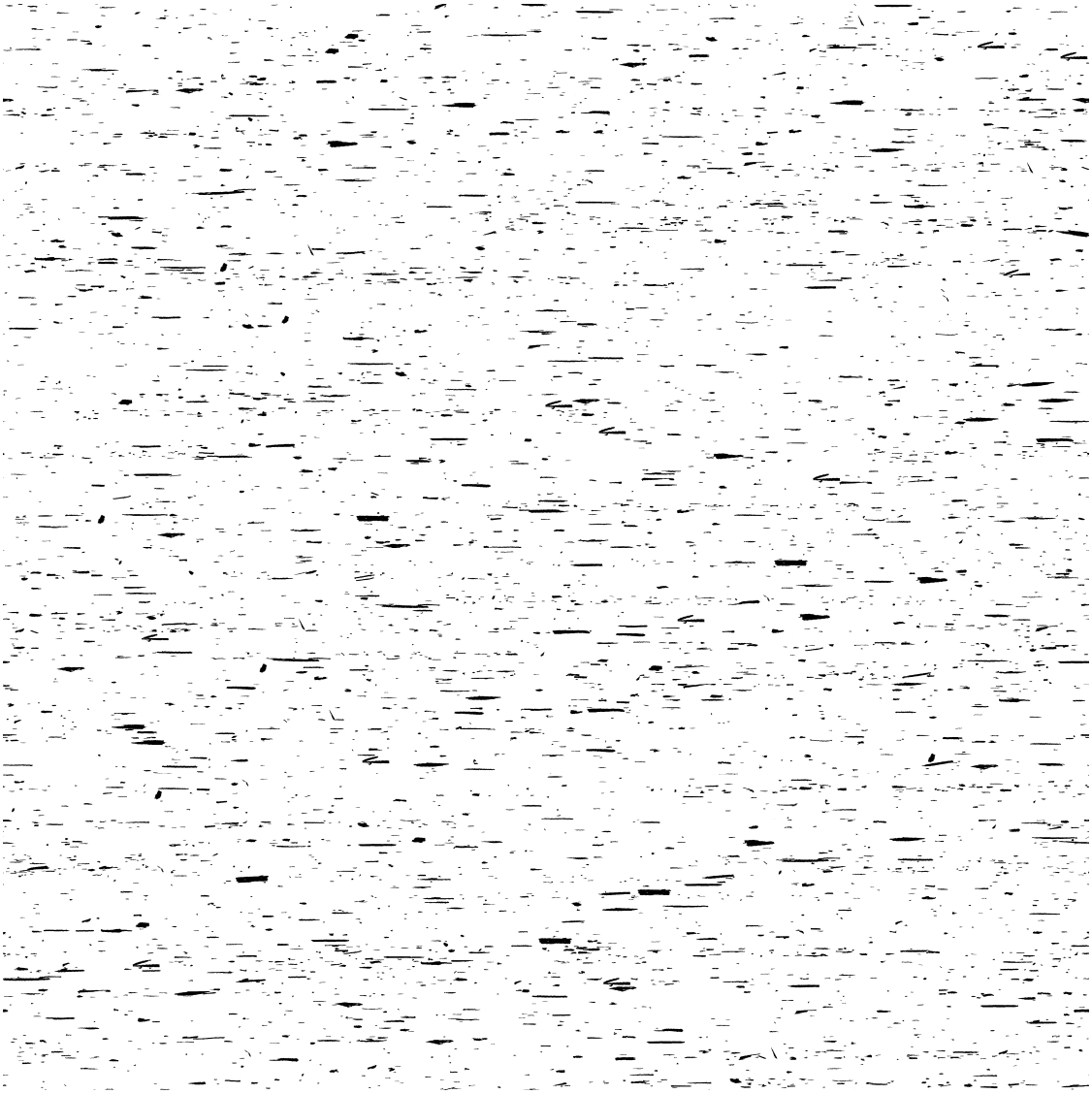


Figure 6.37: (b) Micrograph showing simulated microstructure of extruded boron modified Ti alloy specimen I.

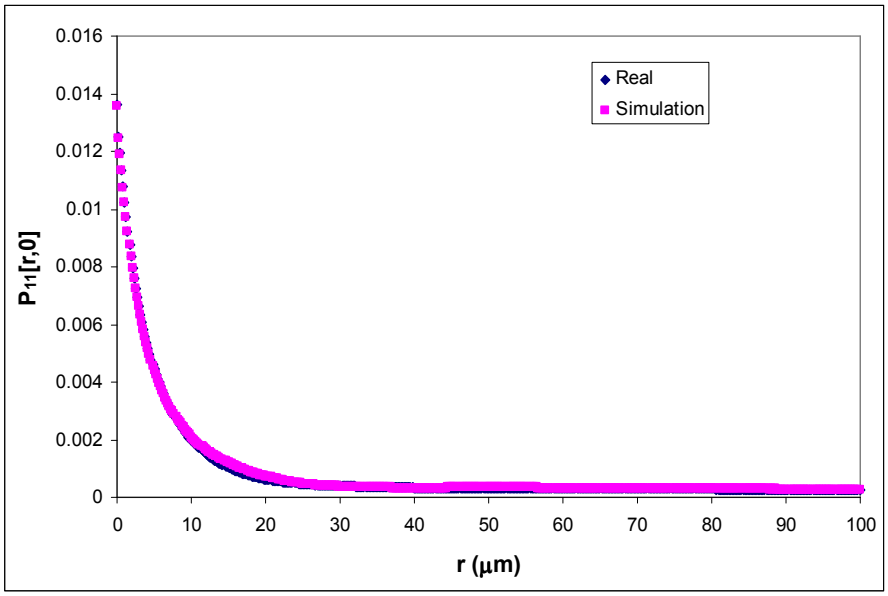



Figure 6.38a: Comparison of two point probability functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen A.

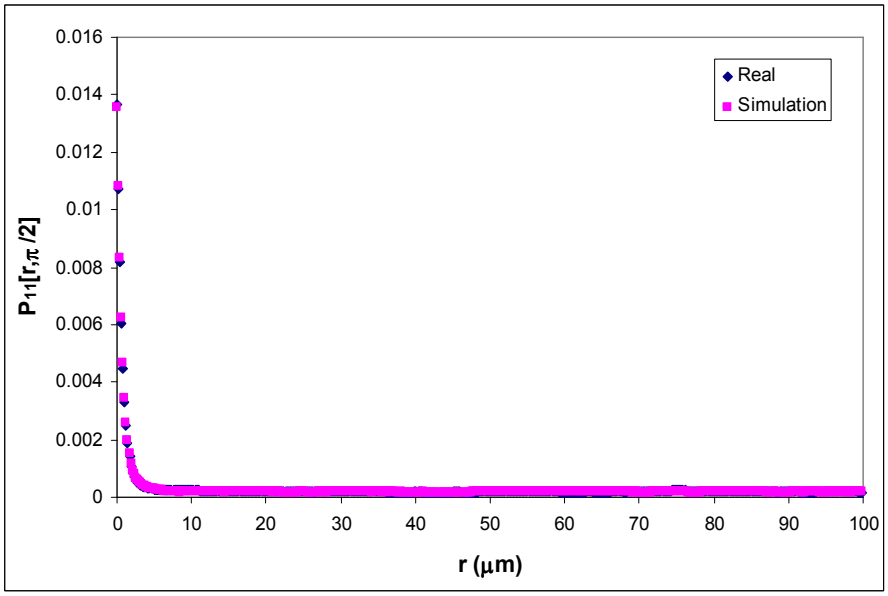



Figure 6.38b: Comparison of two point probability functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen A.

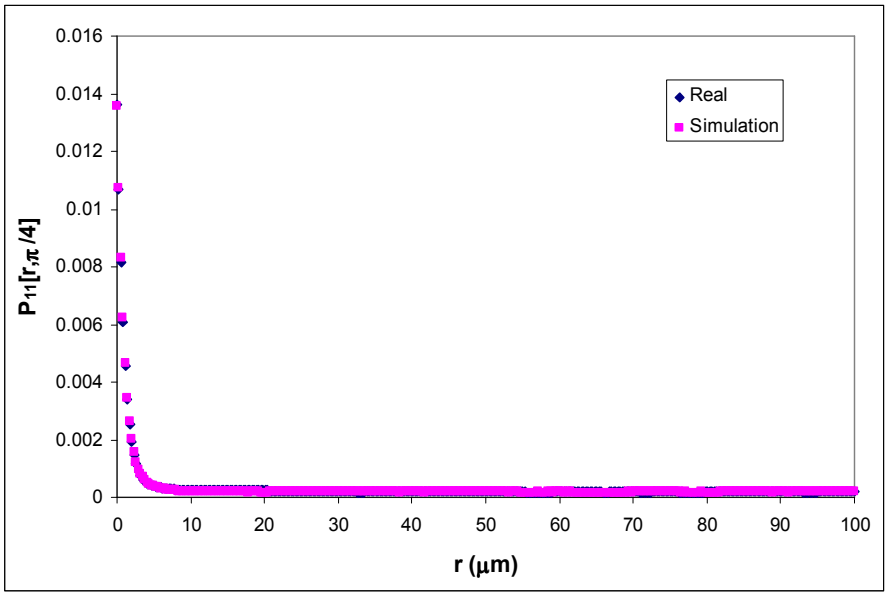



Figure 6.38c: Comparison of two point probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen A.

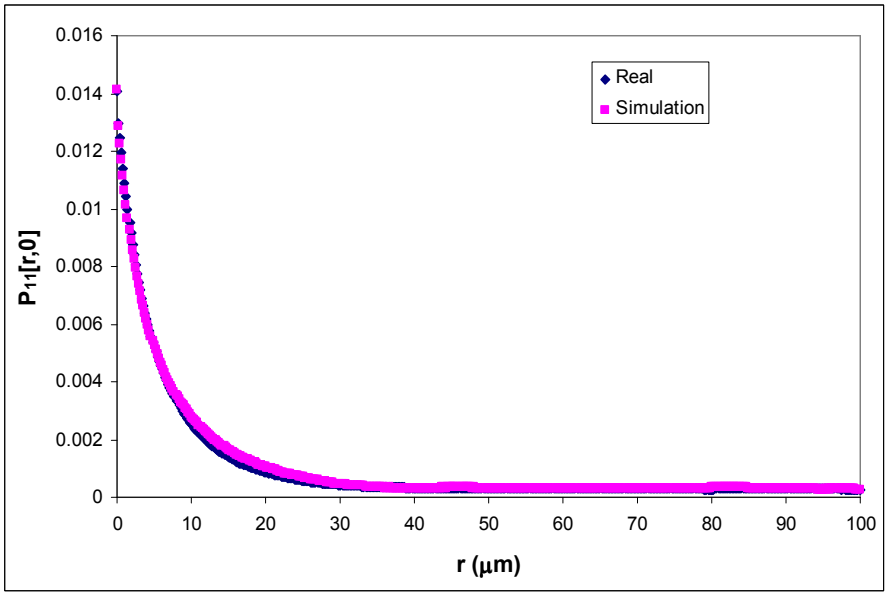



Figure 6.39a: Comparison of two point probability functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen B.

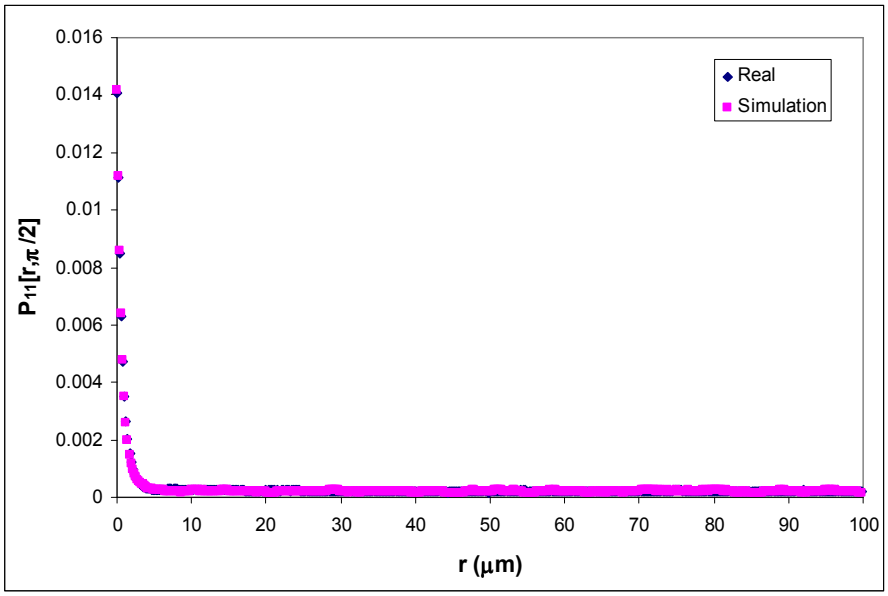



Figure 6.39b: Comparison of two point probability functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen B.

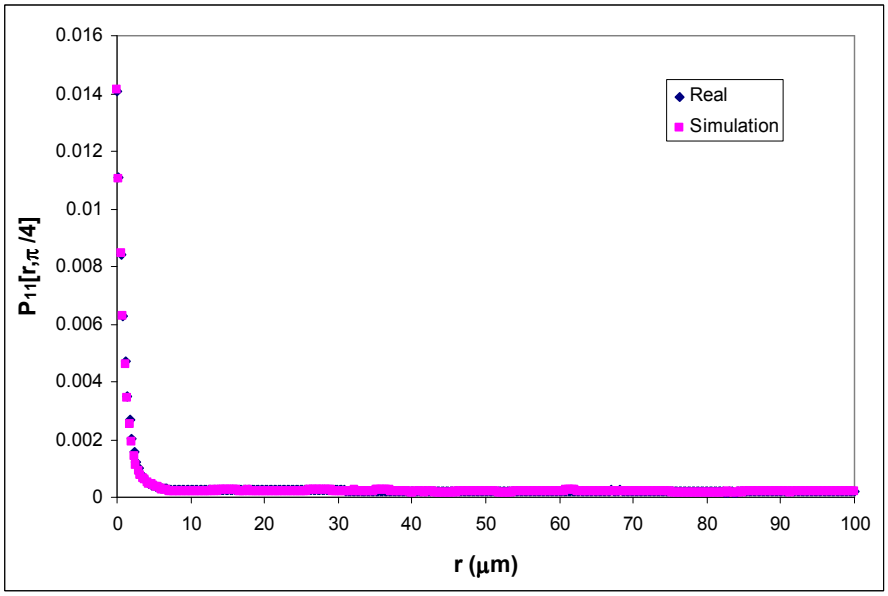



Figure 6.39c: Comparison of two point probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen B.

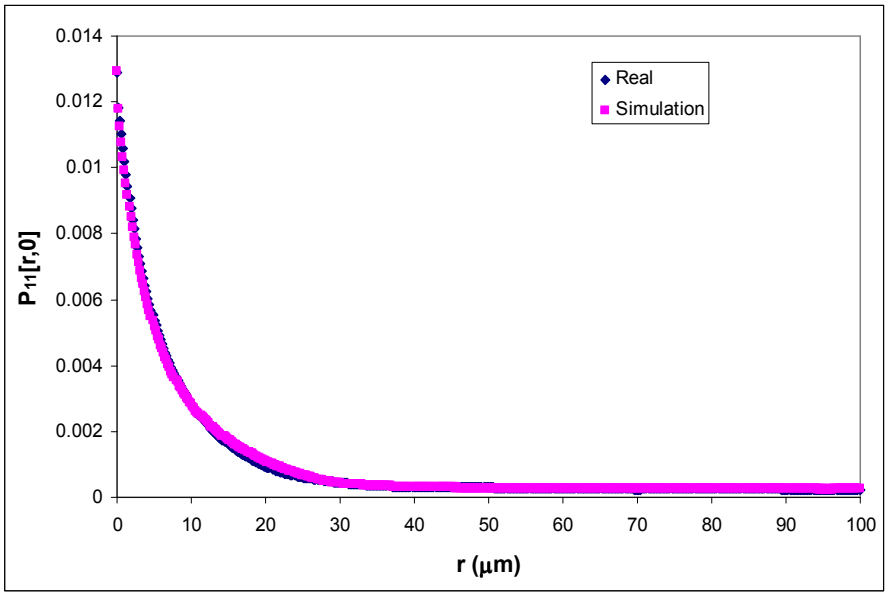



Figure 6.40a: Comparison of two point probability functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen C.

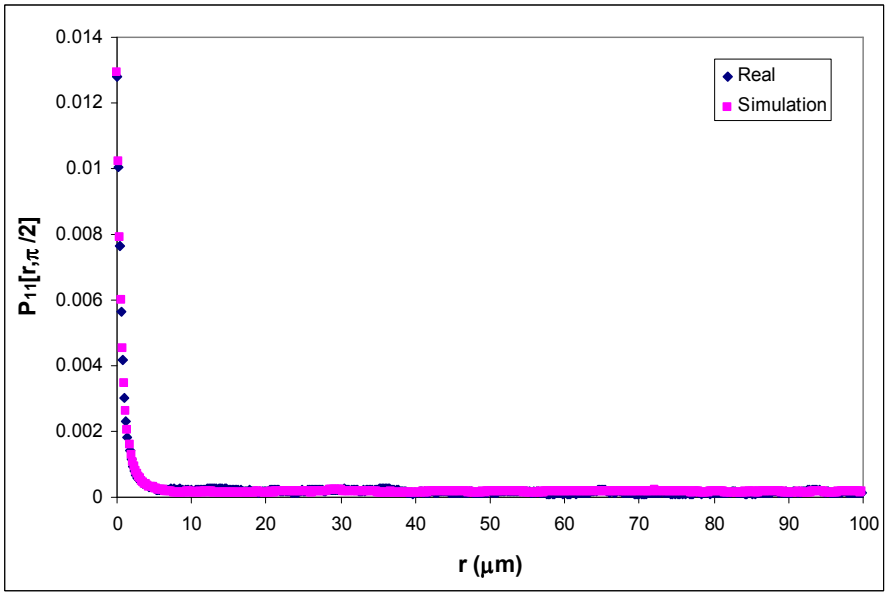



Figure 6.40b: Comparison of two point probability functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen C.

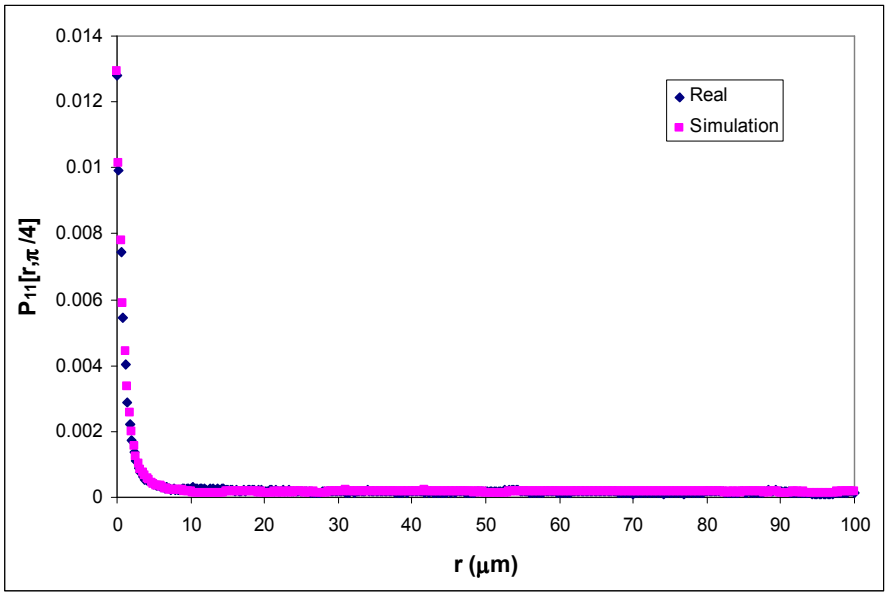



Figure 6.40c: Comparison of two point probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen C.

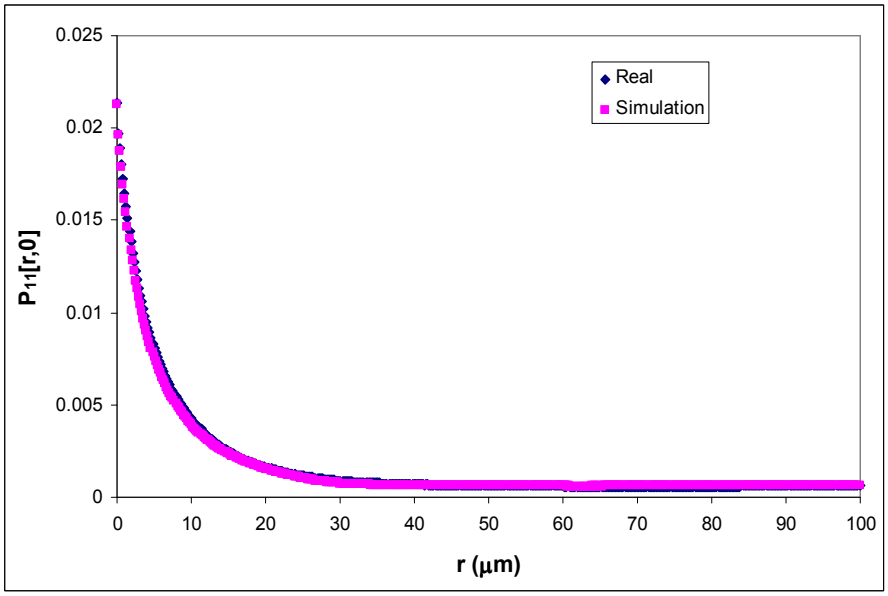



Figure 6.41a: Comparison of two point probability functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen D.

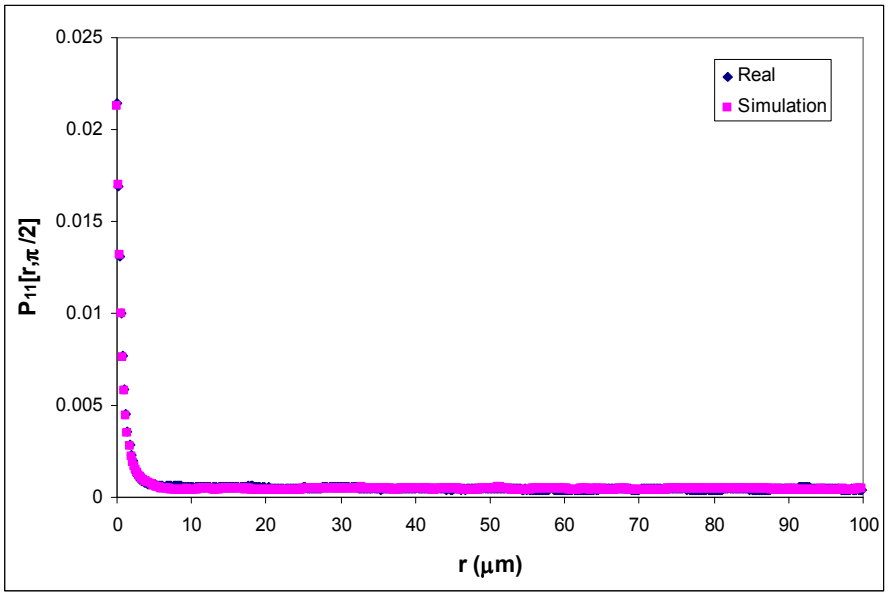



Figure 6.41b: Comparison of two point probability functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen D.

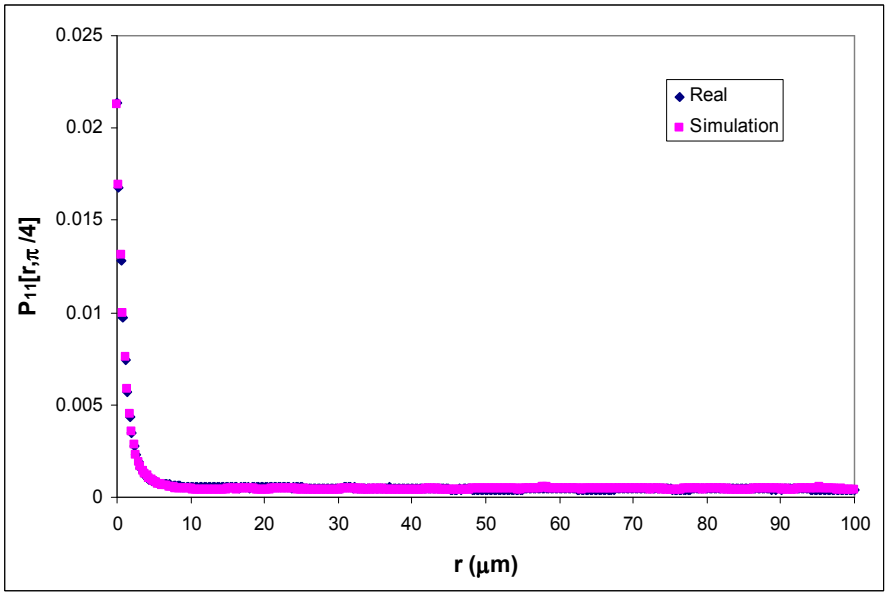



Figure 6.41c: Comparison of two point probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen D.

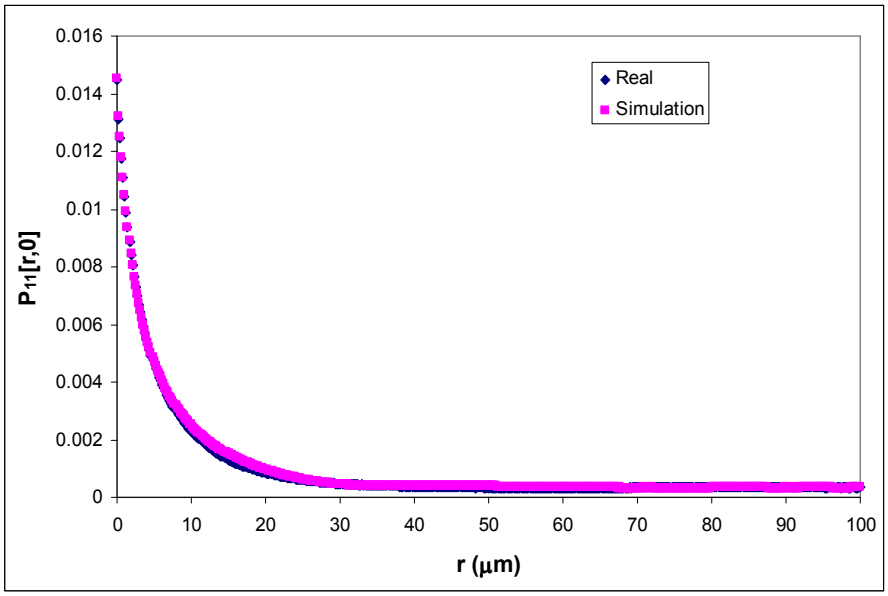



Figure 6.42a: Comparison of two point probability functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen E.

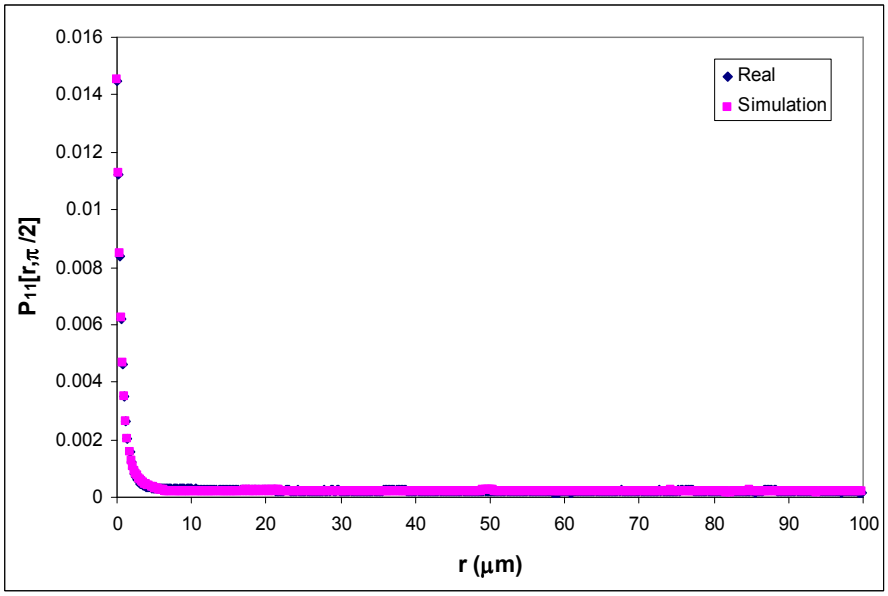



Figure 6.42b: Comparison of two point probability functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen E.

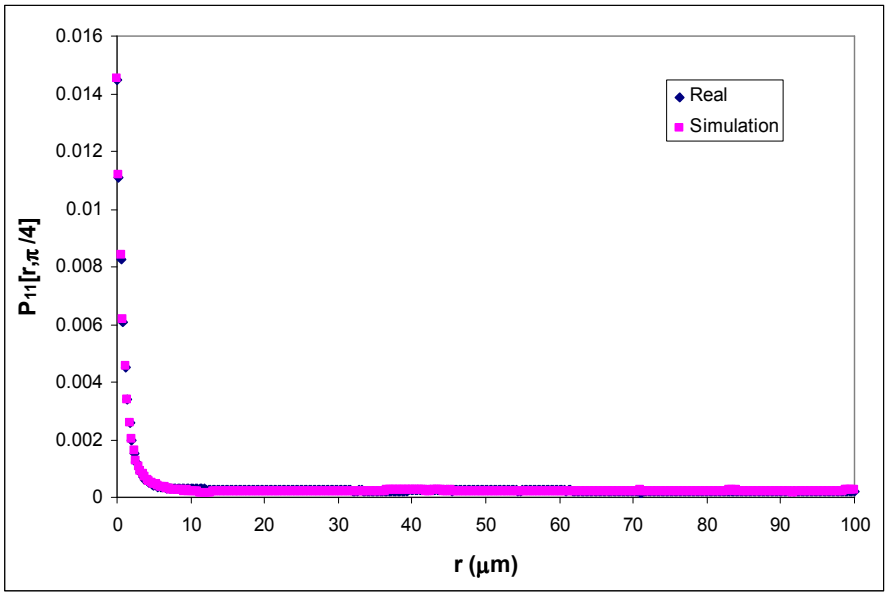



Figure 6.42c: Comparison of two point probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen E.

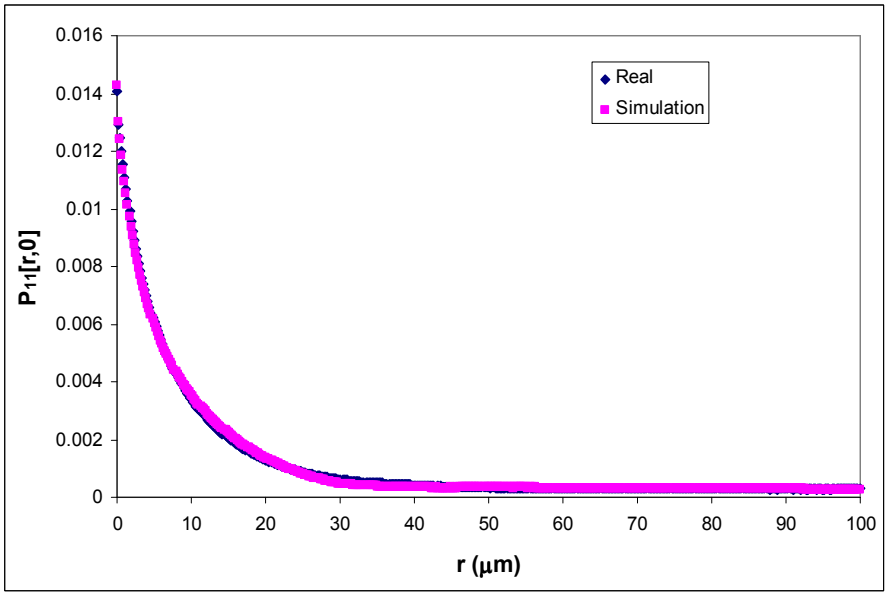



Figure 6.43a: Comparison of two point probability functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen F.

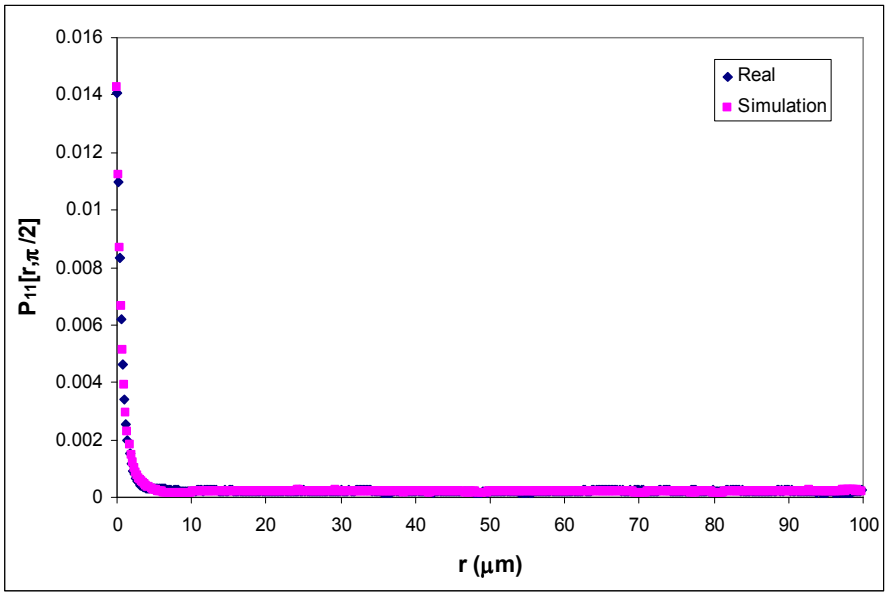



Figure 6.43b: Comparison of two point probability functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen F.

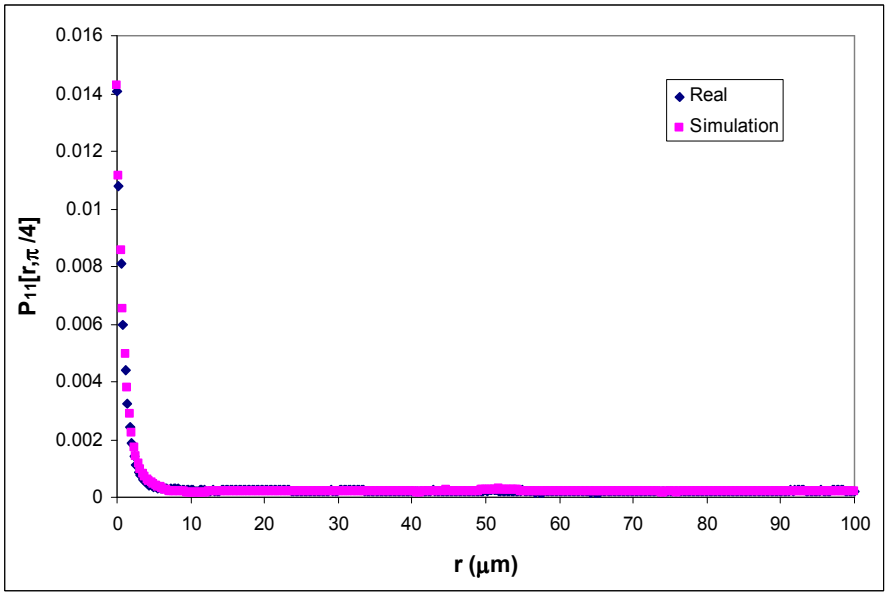



Figure 6.43c: Comparison of two point probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen F.

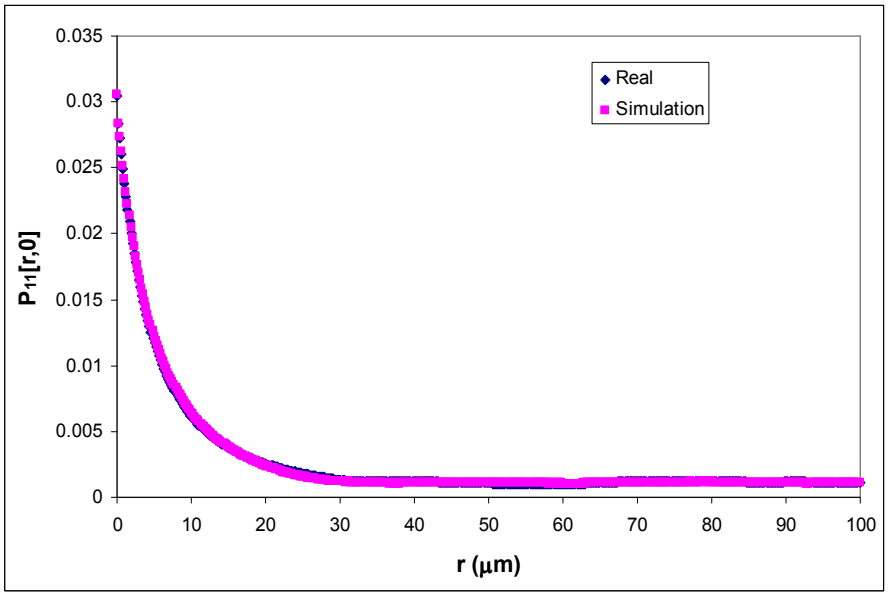



Figure 6.44a: Comparison of two point probability functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen G.

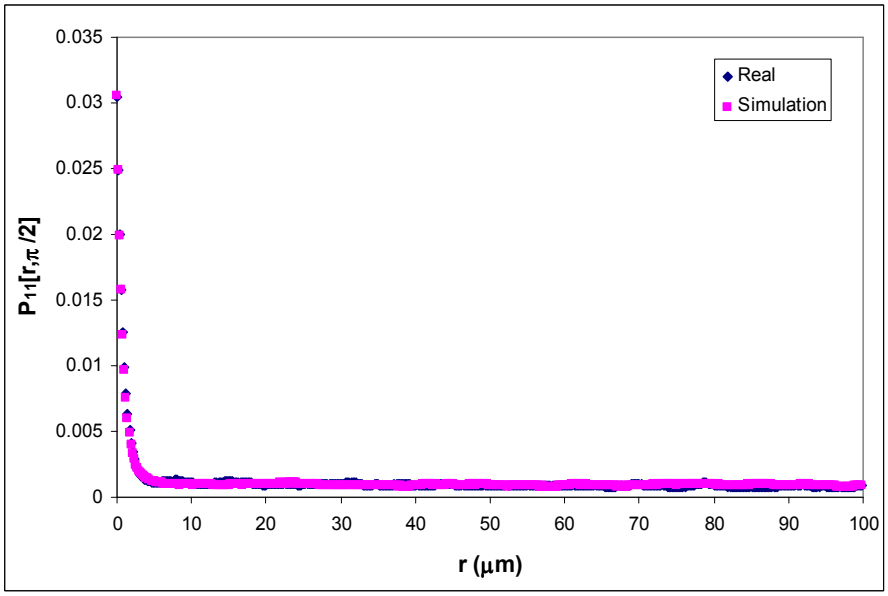



Figure 6.44b: Comparison of two point probability functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen G.

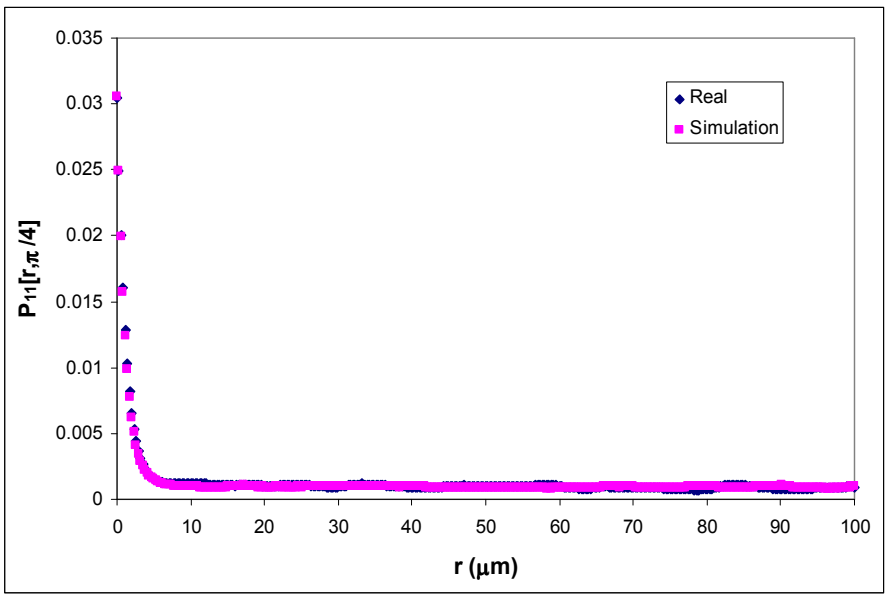



Figure 6.44c: Comparison of two point probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen G.

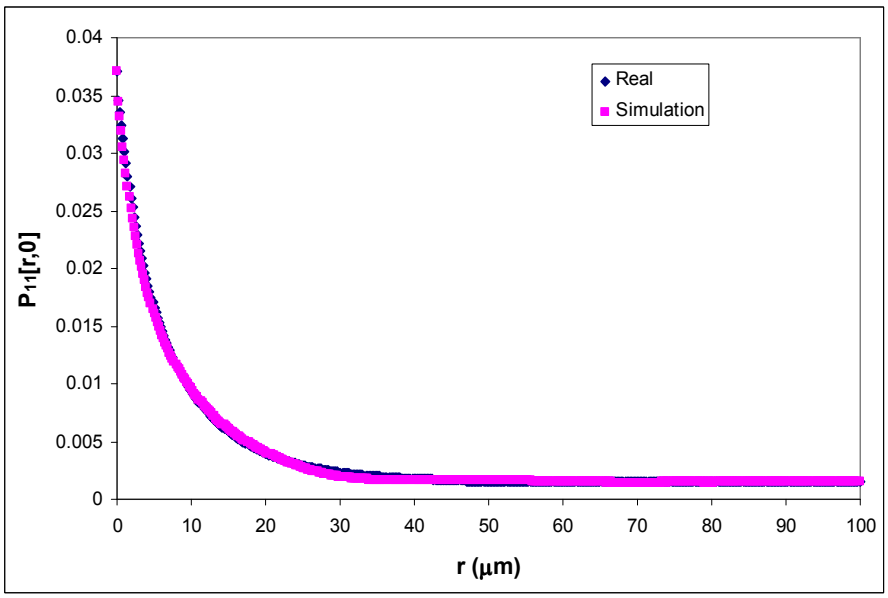



Figure 6.45a: Comparison of two point probability functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen H.

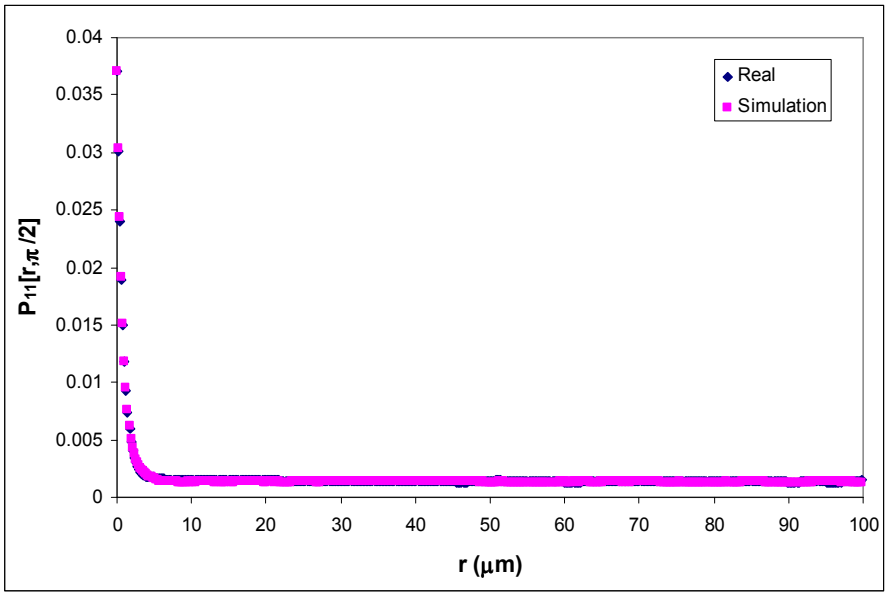



Figure 6.45b: Comparison of two point probability functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen H.

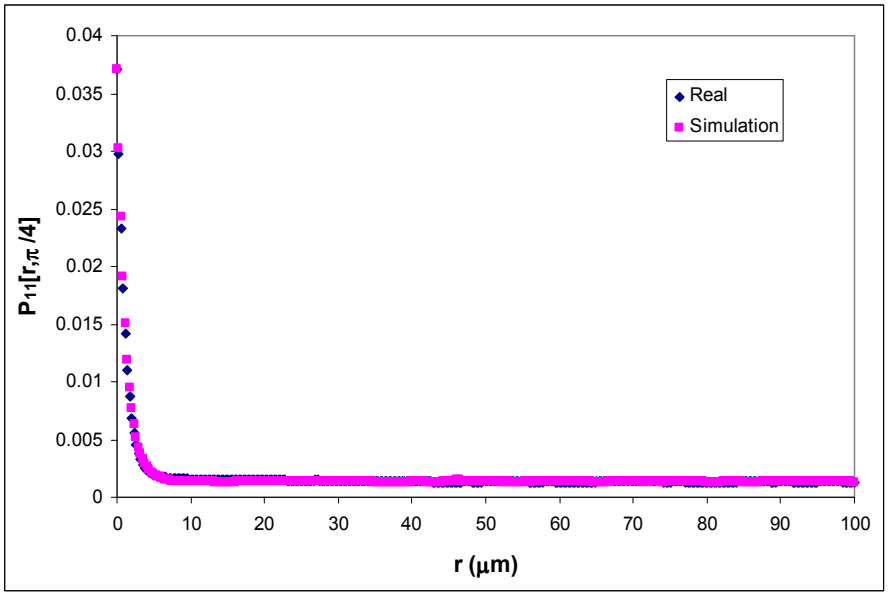



Figure 6.45c: Comparison of two point probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen H.

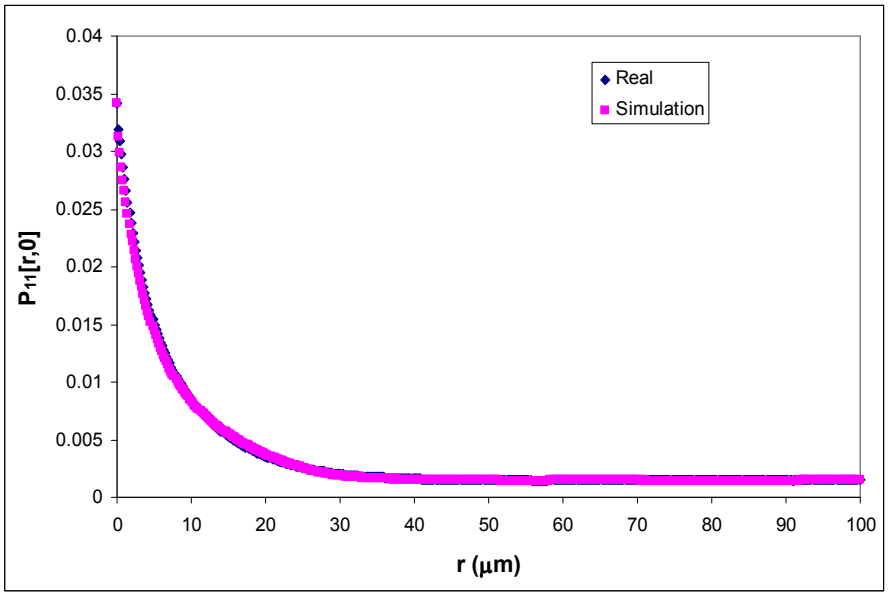



Figure 6.46a: Comparison of two point probability functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen I.

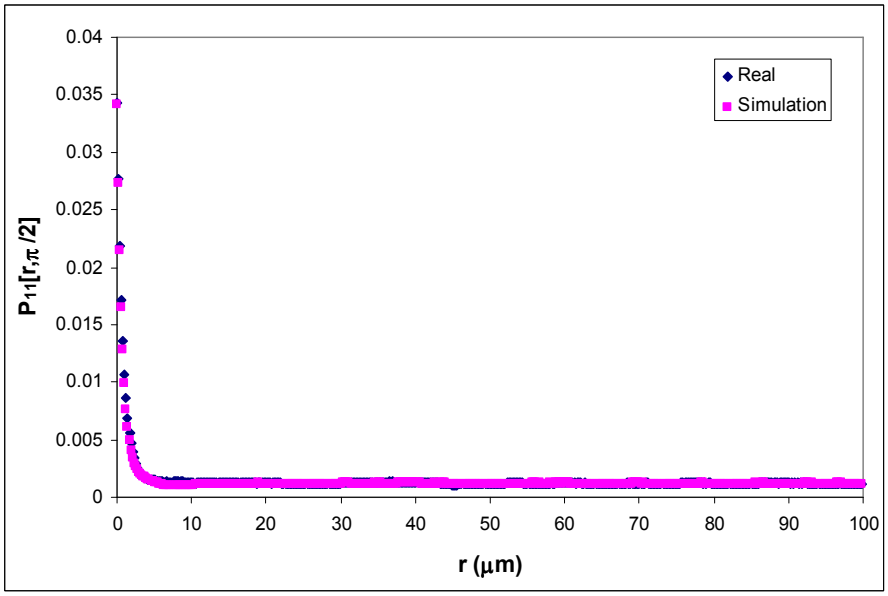



Figure 6.46b: Comparison of two point probability functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen I.

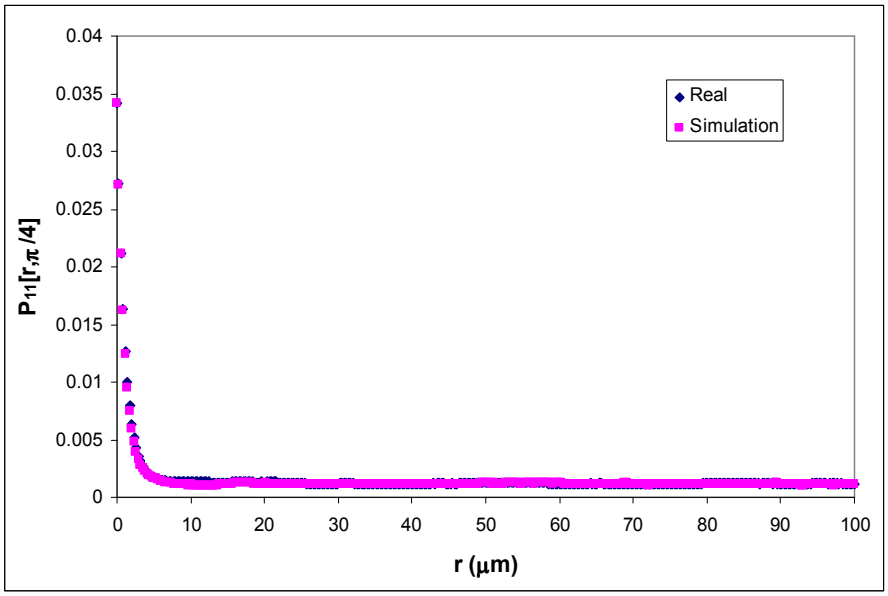



Figure 6.46c: Comparison of two point probability functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen I.

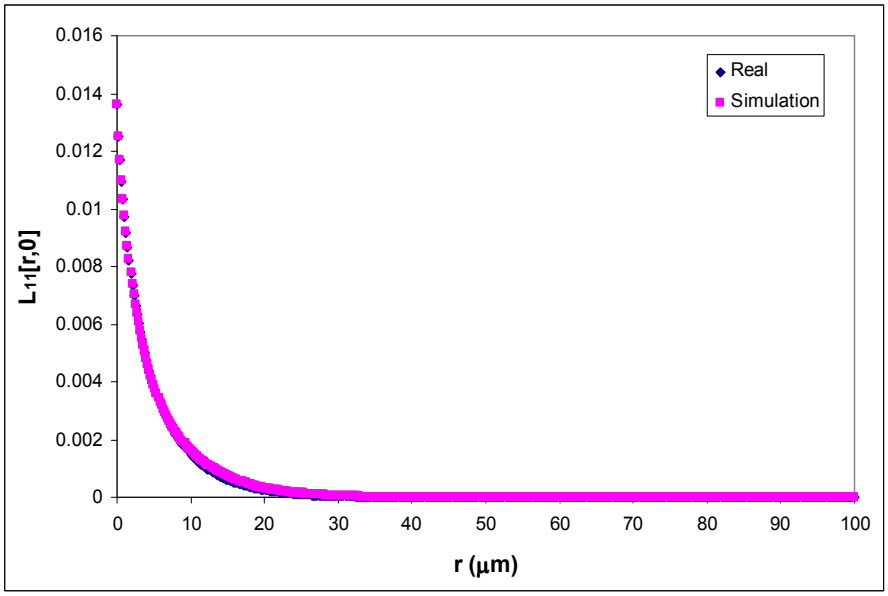



Figure 6.47a: Comparison of lineal path distribution functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen A.

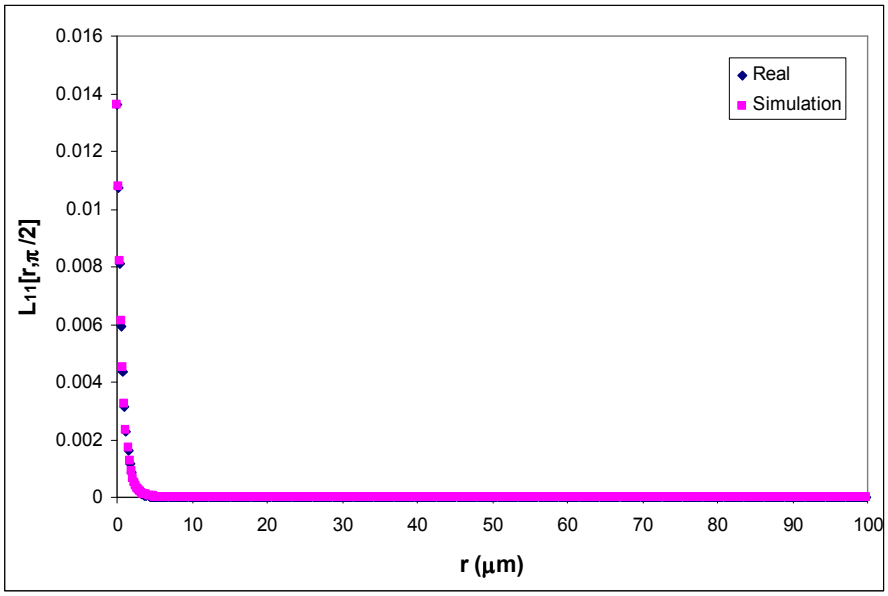



Figure 6.47b: Comparison of lineal path distribution functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen A.

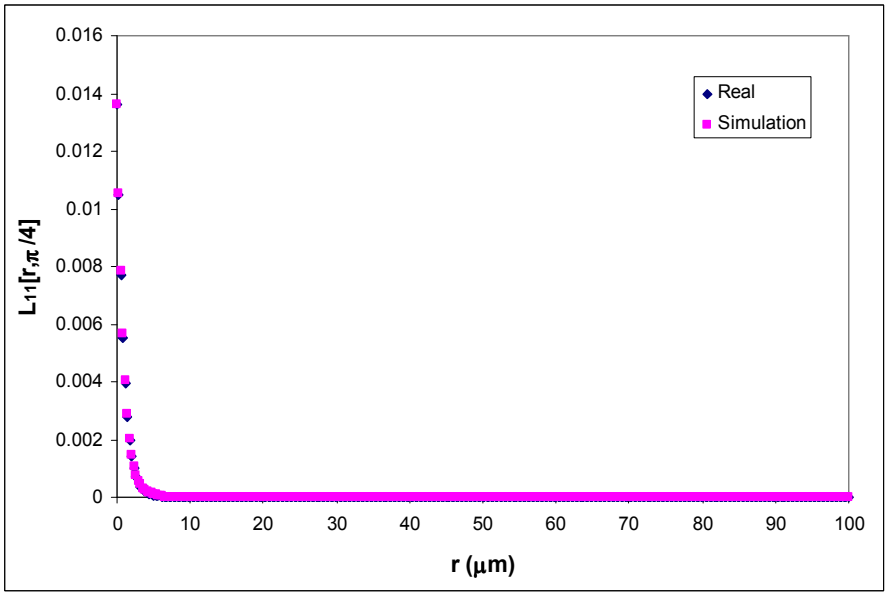



Figure 6.47c: Comparison of lineal path distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen A.

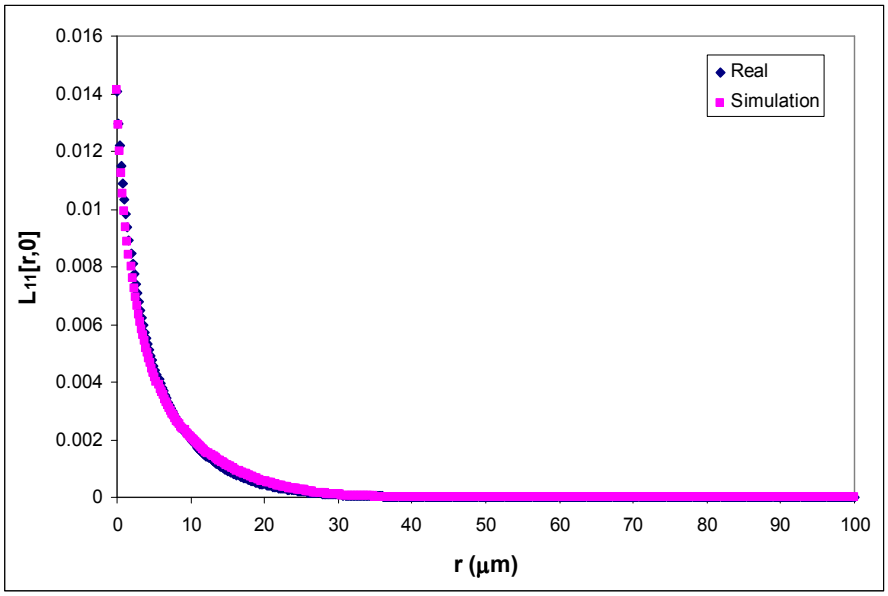



Figure 6.48a: Comparison of lineal path distribution functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen B.

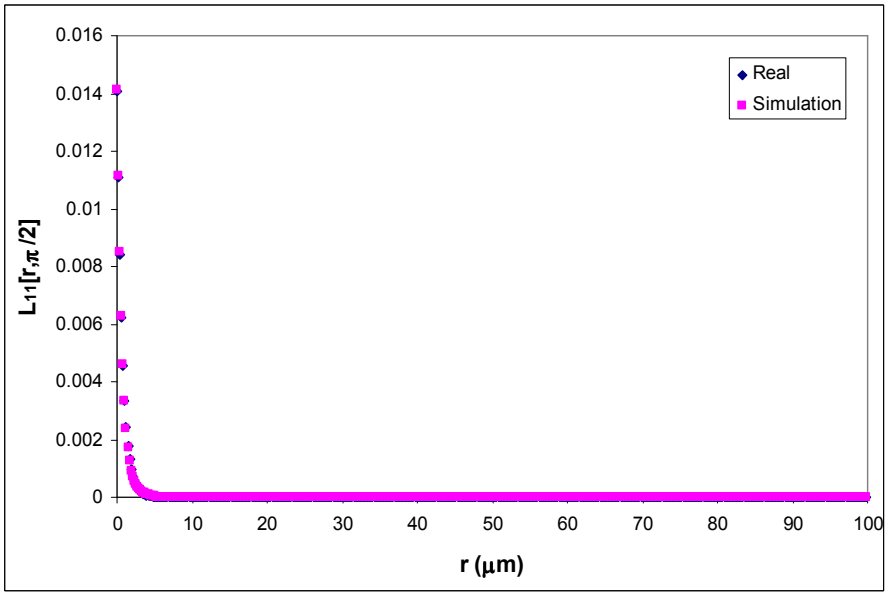



Figure 6.48b: Comparison of lineal path distribution functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen B.

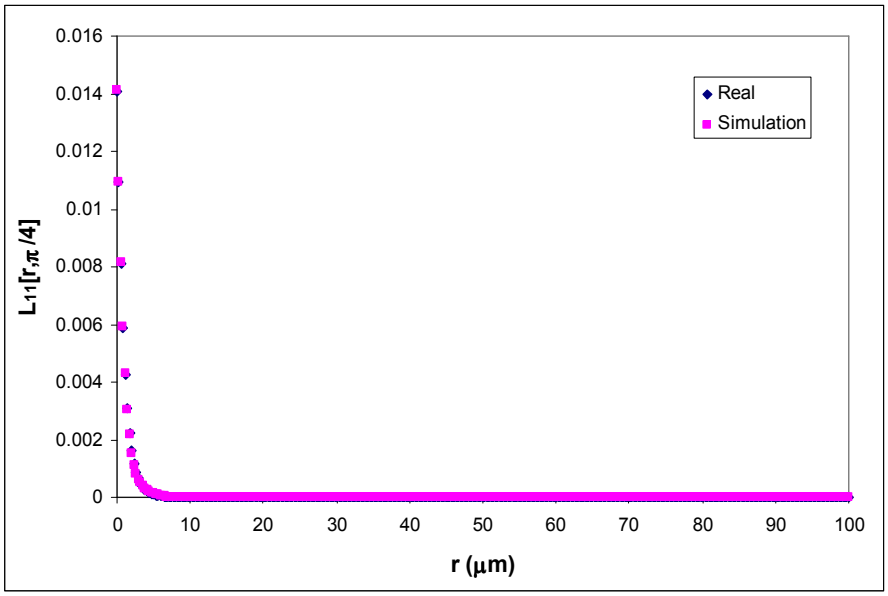



Figure 6.48c: Comparison of lineal path distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen B.

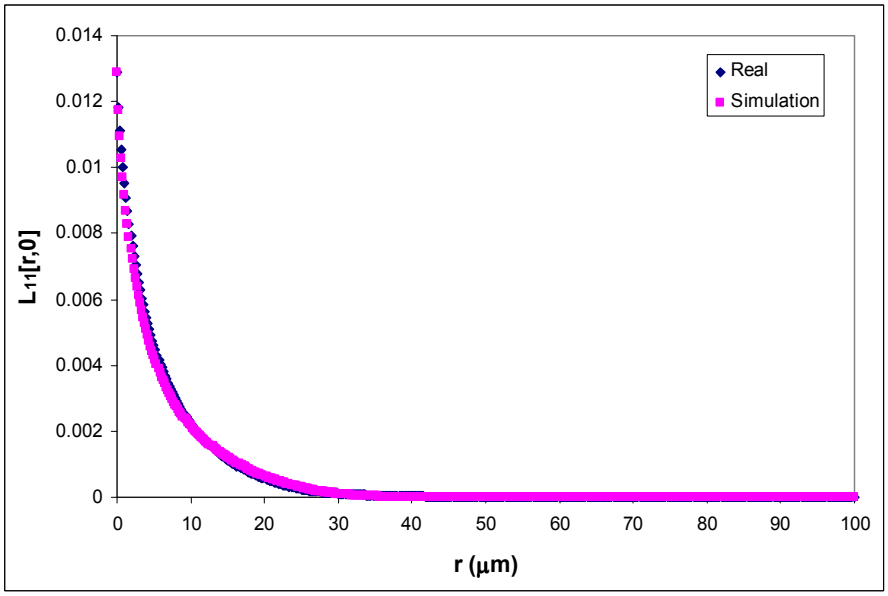



Figure 6.49a: Comparison of lineal path distribution functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen C.

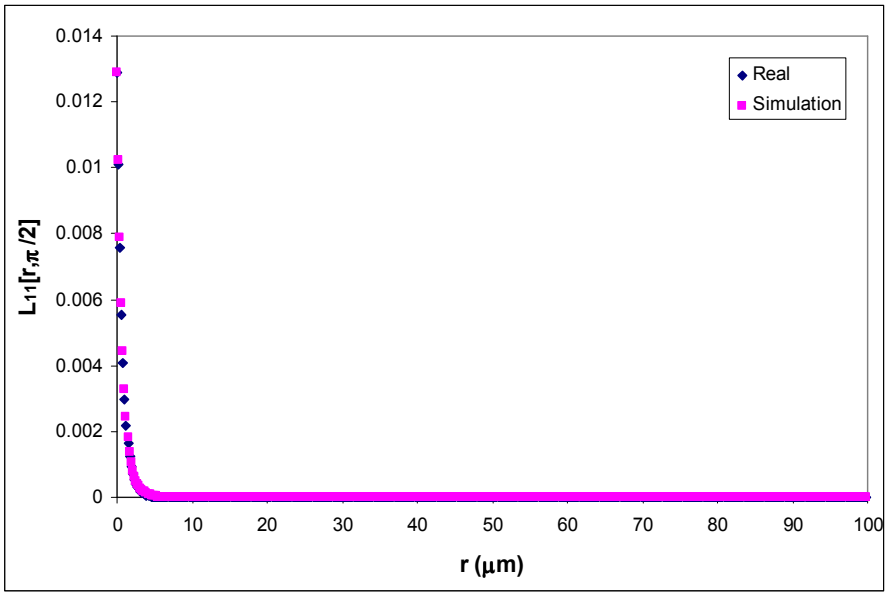



Figure 6.49b: Comparison of lineal path distribution functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen C.

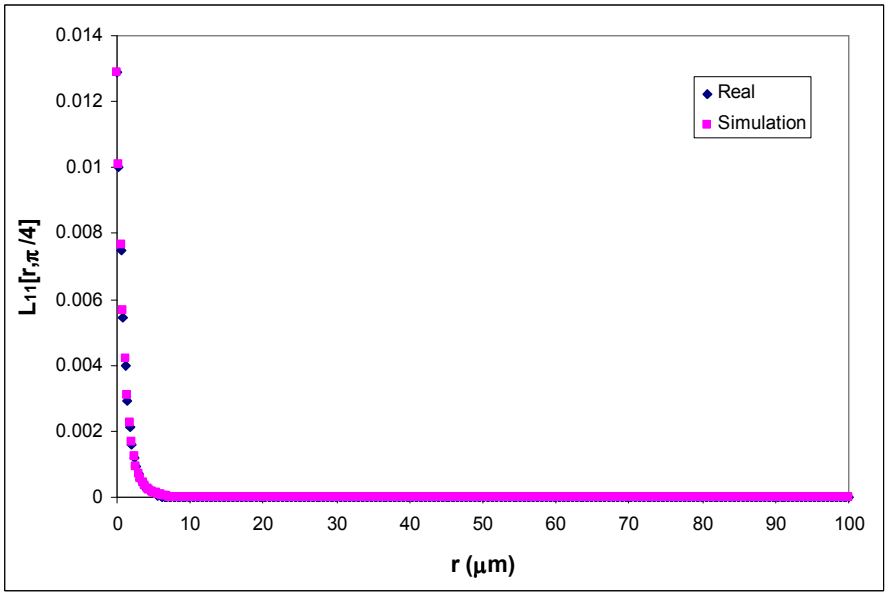



Figure 6.49c: Comparison of lineal path distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen C.

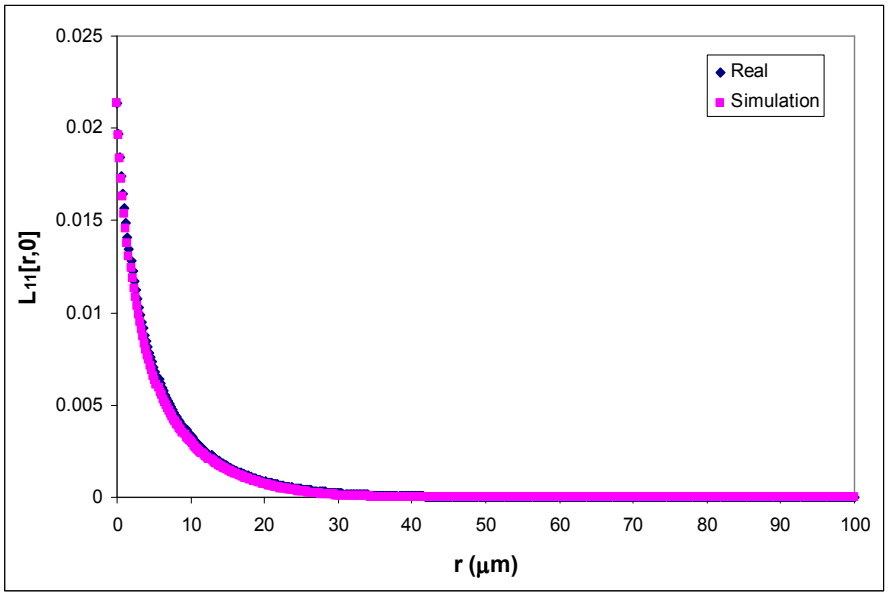



Figure 6.50a: Comparison of lineal path distribution functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen D.

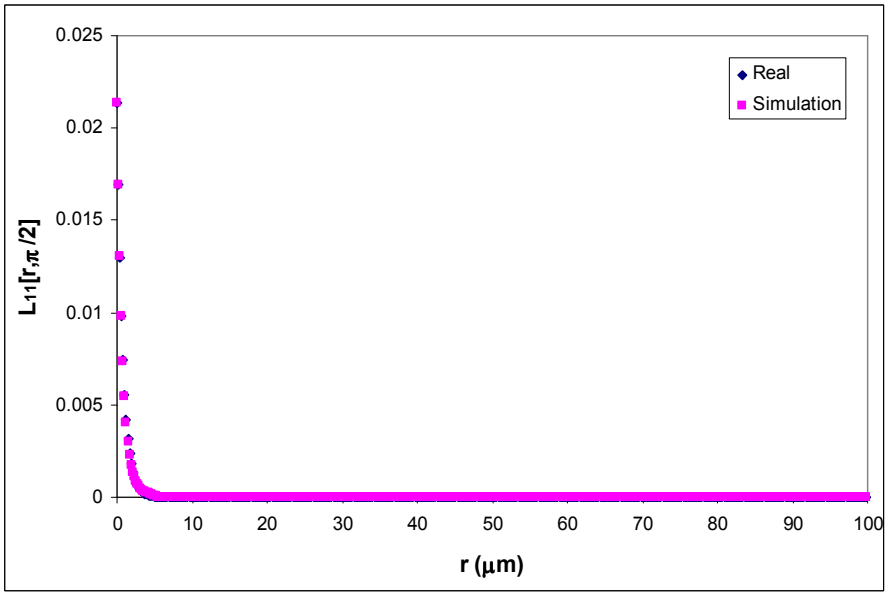



Figure 6.50b: Comparison of lineal path distribution functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen D.

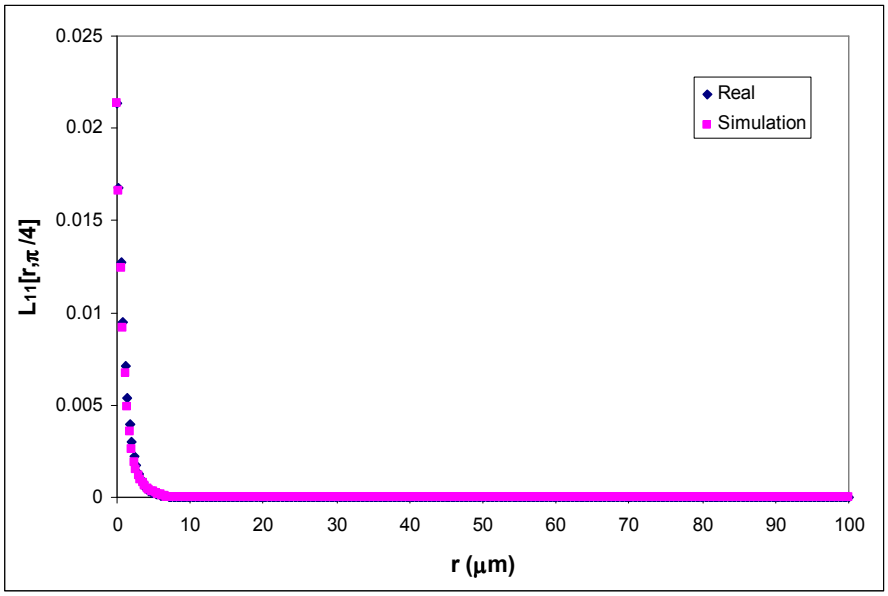



Figure 6.50c: Comparison of lineal path distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen D.

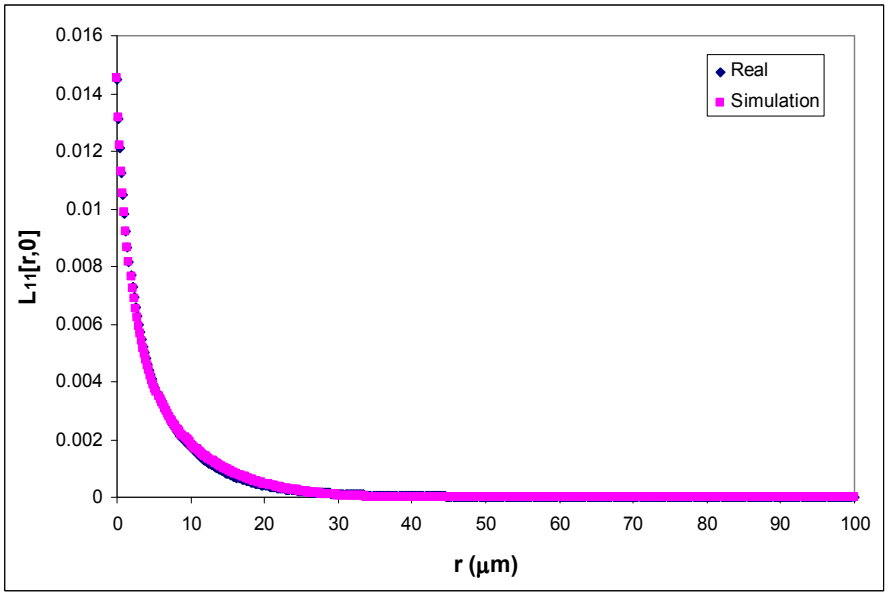



Figure 6.51a: Comparison of lineal path distribution functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen E.

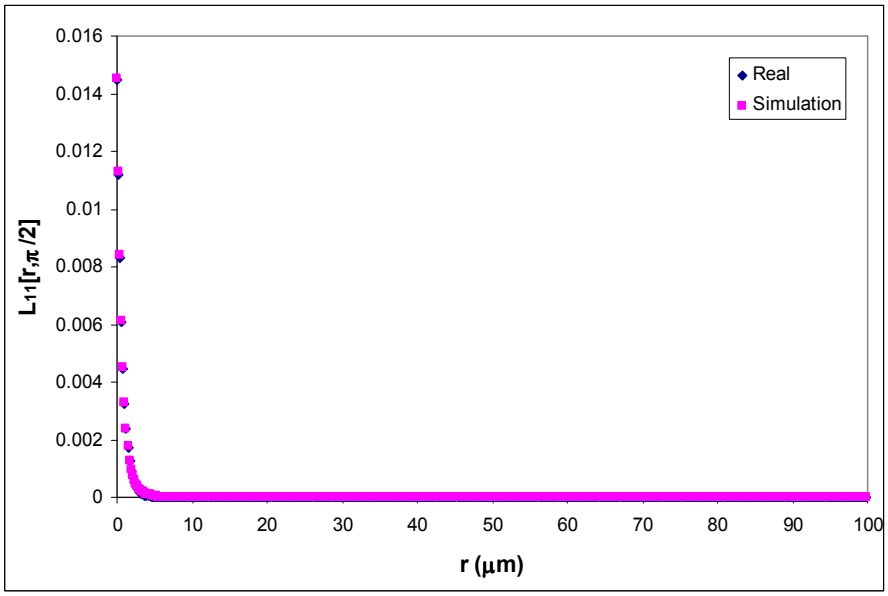



Figure 6.51b: Comparison of lineal path distribution functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen E.

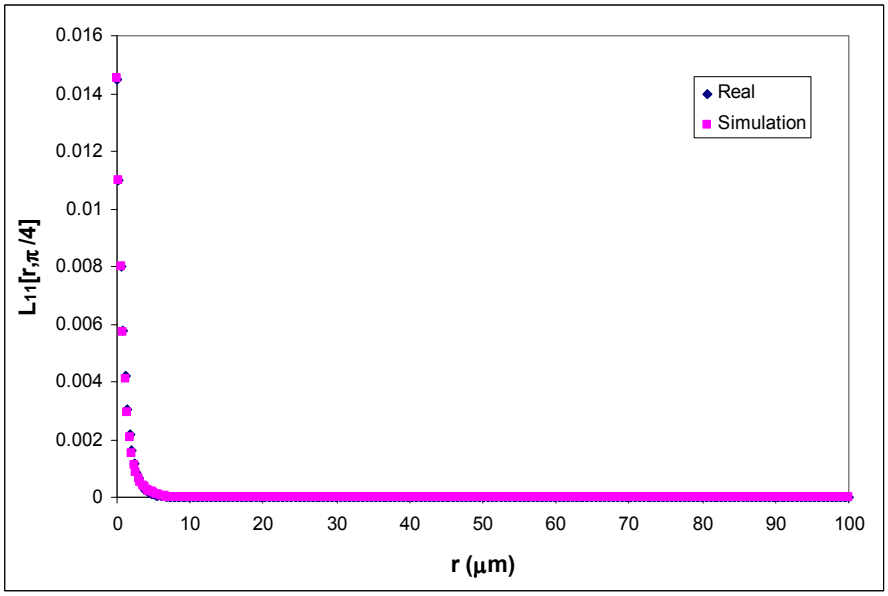



Figure 6.51c: Comparison of lineal path distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen E.

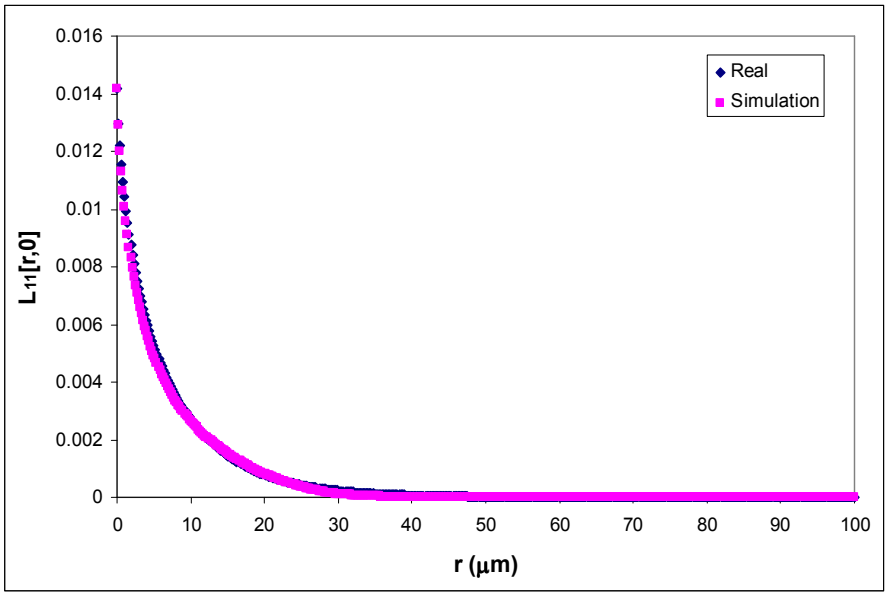



Figure 6.52a: Comparison of lineal path distribution functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen F.

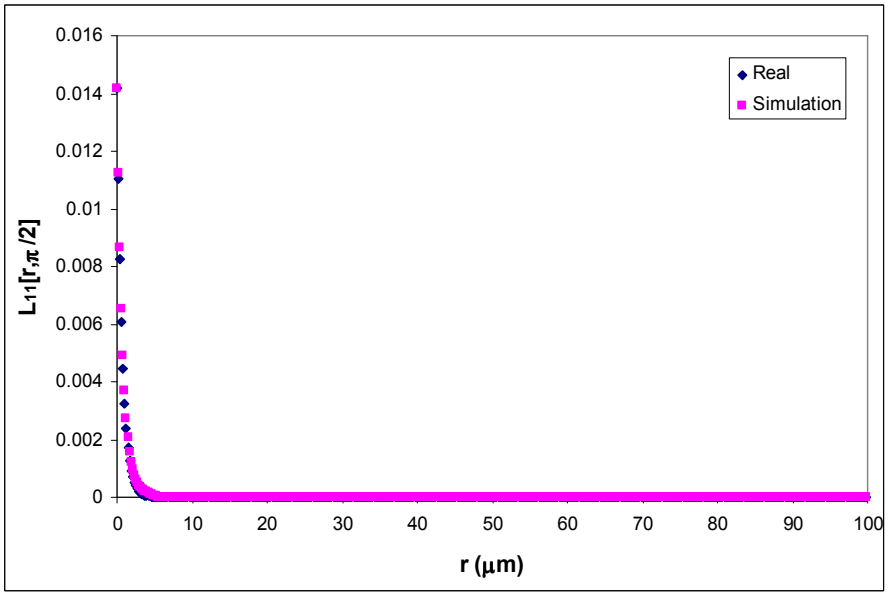



Figure 6.52b: Comparison of lineal path distribution functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen F.

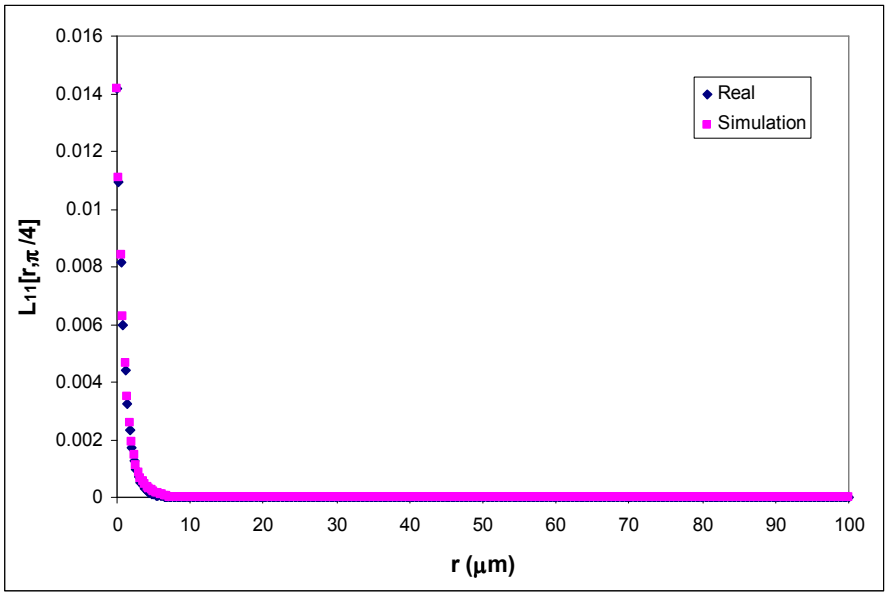



Figure 6.52c: Comparison of lineal path distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen F.

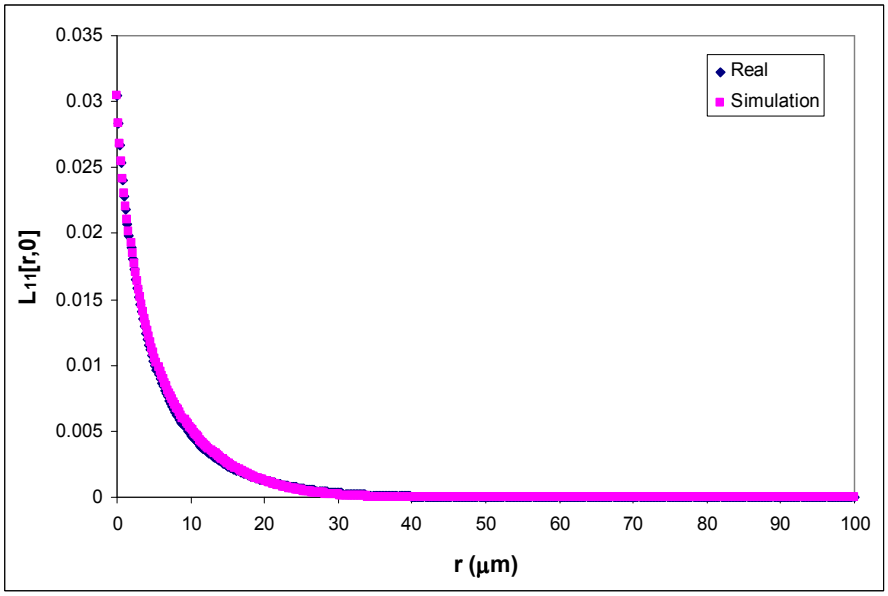



Figure 6.53a: Comparison of lineal path distribution functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen G.

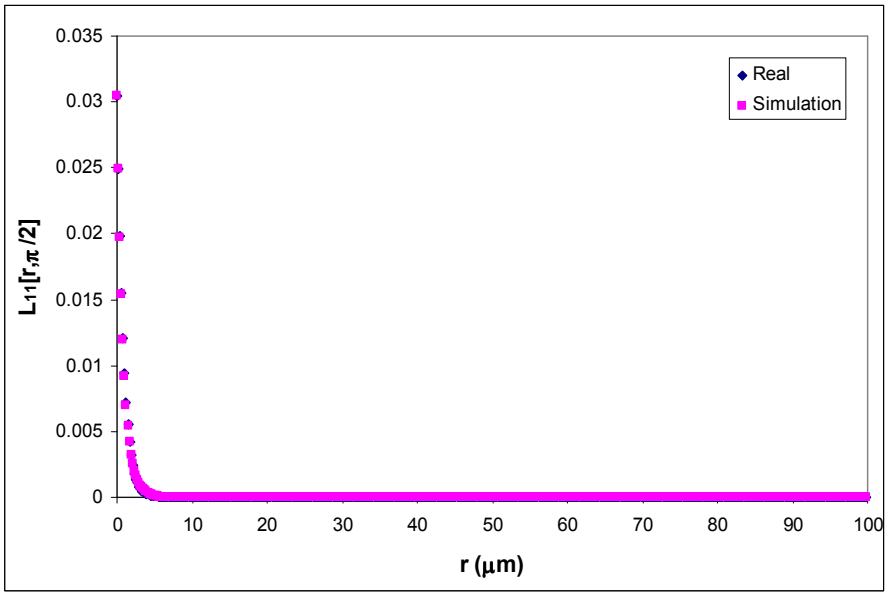



Figure 6.53b: Comparison of lineal path distribution functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen G.

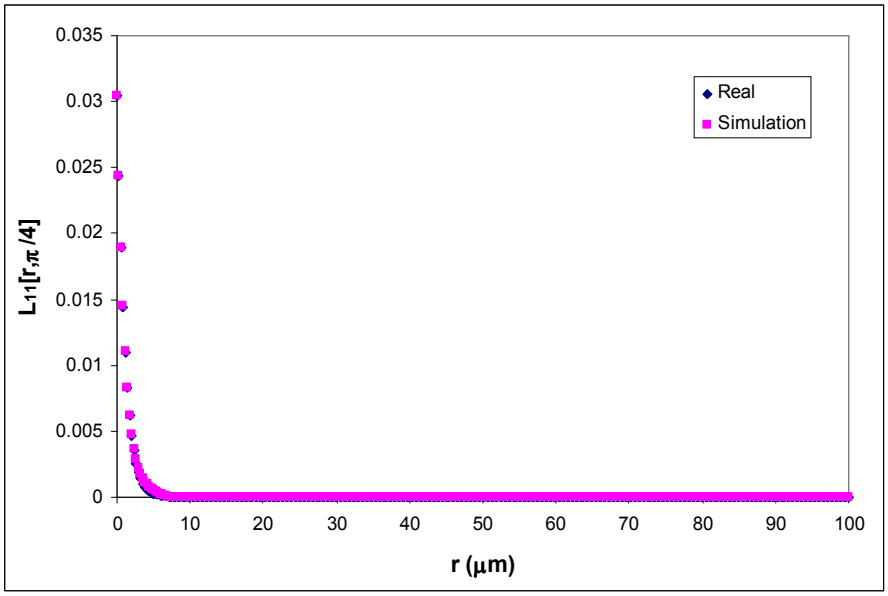



Figure 6.53c: Comparison of lineal path distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen G.

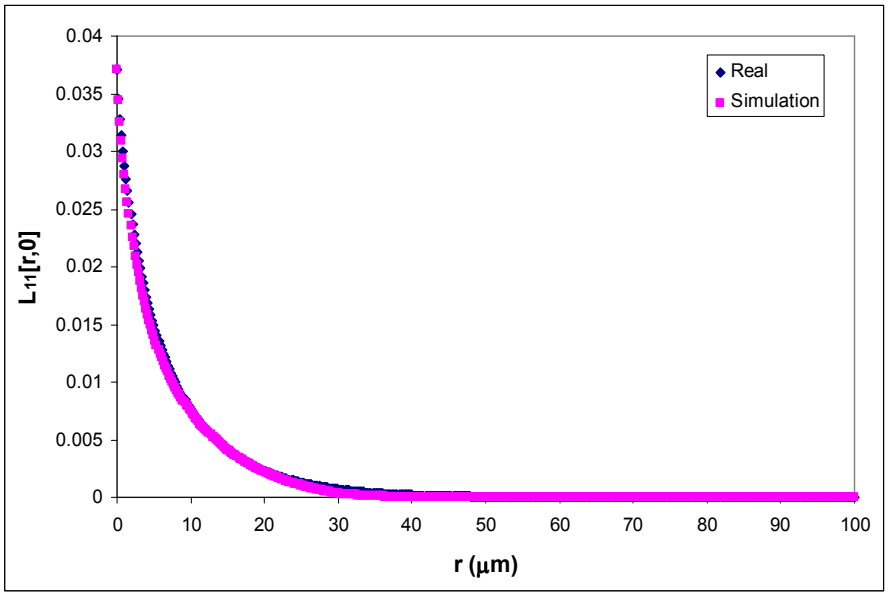



Figure 6.54a: Comparison of lineal path distribution functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen H.

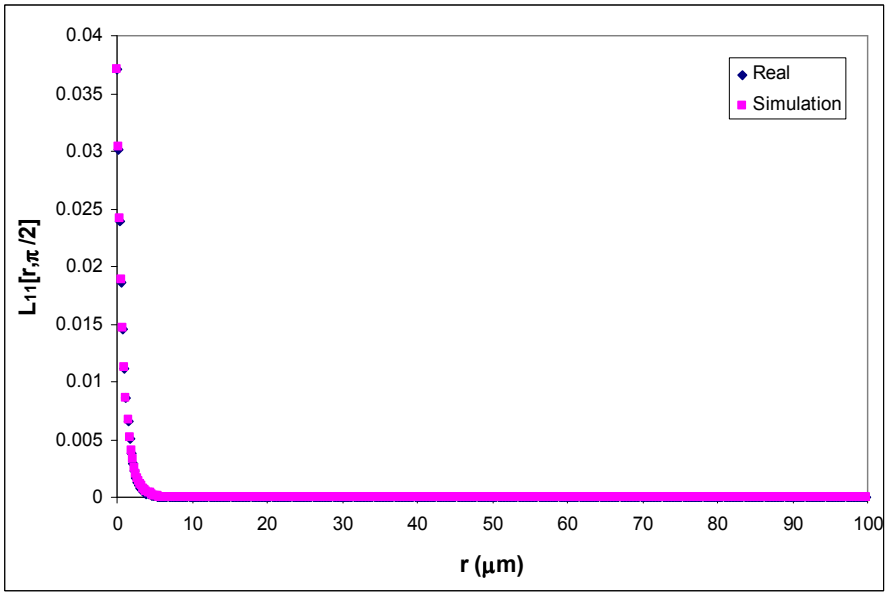



Figure 6.54b: Comparison of lineal path distribution functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen H.

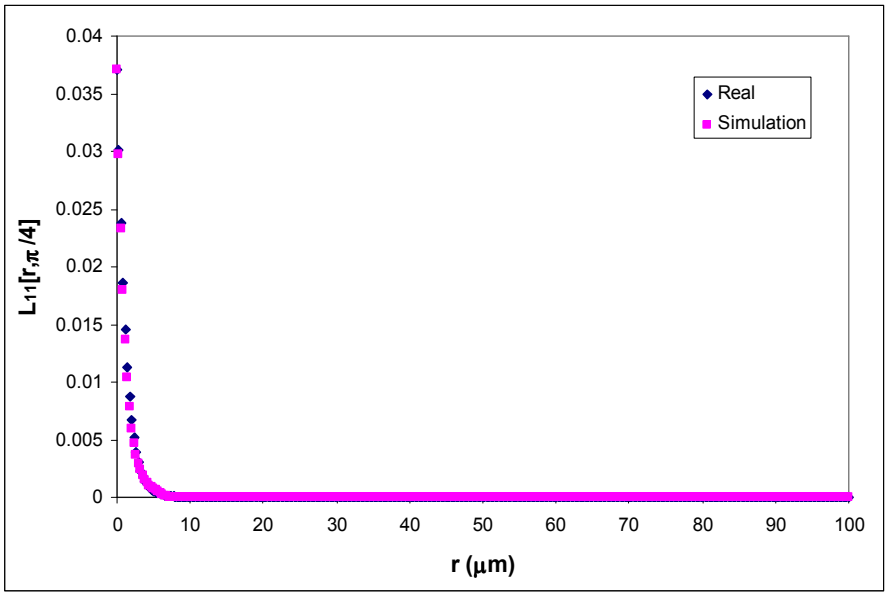



Figure 6.54c: Comparison of lineal path distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen H.

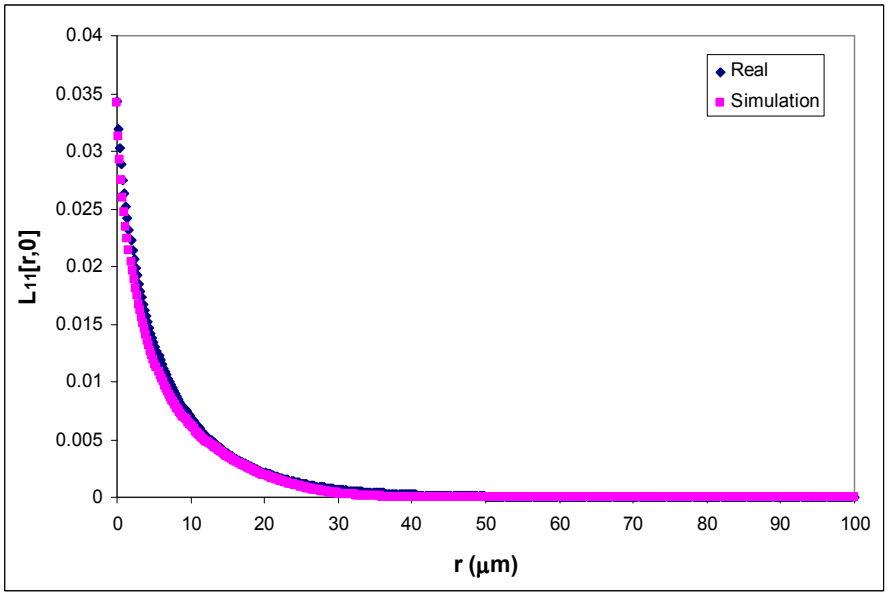



Figure 6.55a: Comparison of lineal path distribution functions measured in the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen I.

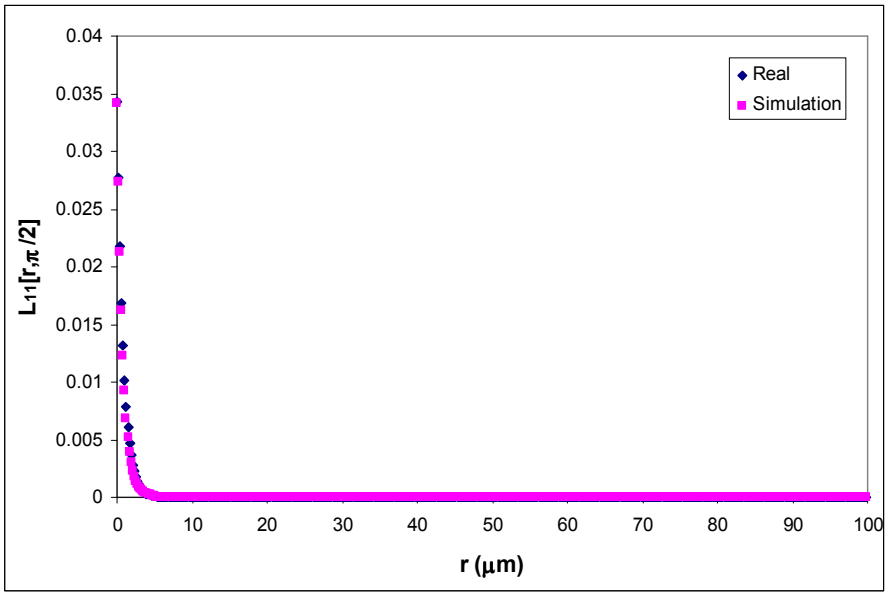



Figure 6.55b: Comparison of lineal path distribution functions measured in the transverse direction for real and simulated microstructures of extruded boron modified Ti alloy specimen I.

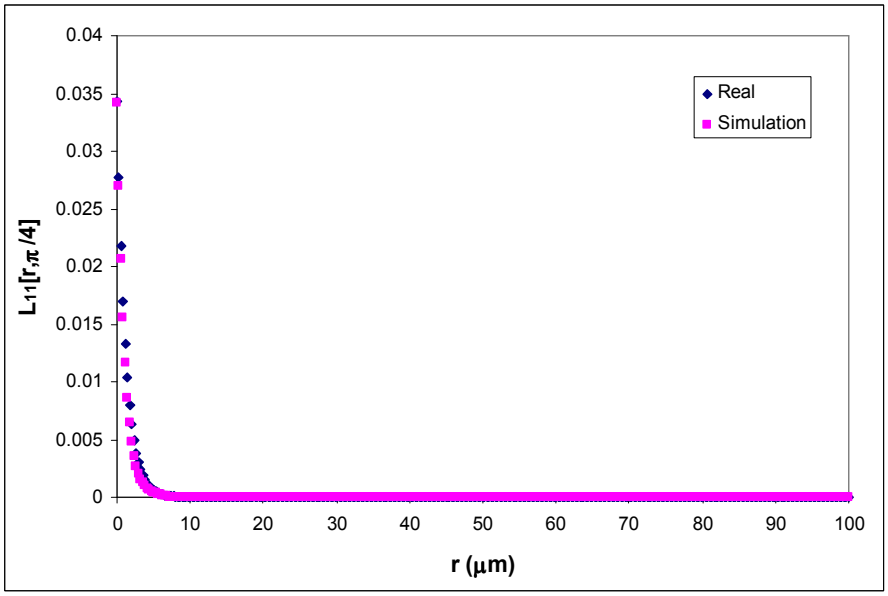



Figure 6.55c: Comparison of lineal path distribution functions measured at an angle of 45° from the extrusion direction for real and simulated microstructures of extruded boron modified Ti alloy specimen I.

Table 6.3: Values of Simulation parameters used to generate simulated microstructures of extruded boron modified Ti-6Al-4V alloys.

| Specimen ID | Cluster Major Axis Length (μm) | Cluster Axial Ratio | Cluster Number Density per $mm^2$ | Cluster Intensity | Rotation Angle |
|---|---|---|---|---|---|
| A | 355.3 | 42.5 | 51.28 | 5 | 0 |
| B | 418 | 57.14 | 50.36 | 5 | 4 |
| C | 459.8 | 62.85 | 45.78 | 5 | 10 |
| | | | | | |
| D | 271.7 | 32.5 | 78.75 | 4 | 0.5 |
| E | 402.32 | 48.12 | 54.94 | 4.5 | 2 |
| F | 495.33 | 59.25 | 45.78 | 4 | 10 |
| | | | | | |
| G | 209 | 25 | 116.29 | 3 | 0 |
| H | 334.4 | 40 | 73.25 | 2.5 | 2 |
| I | 522.5 | 62.5 | 46.7 | 3 | 10 |

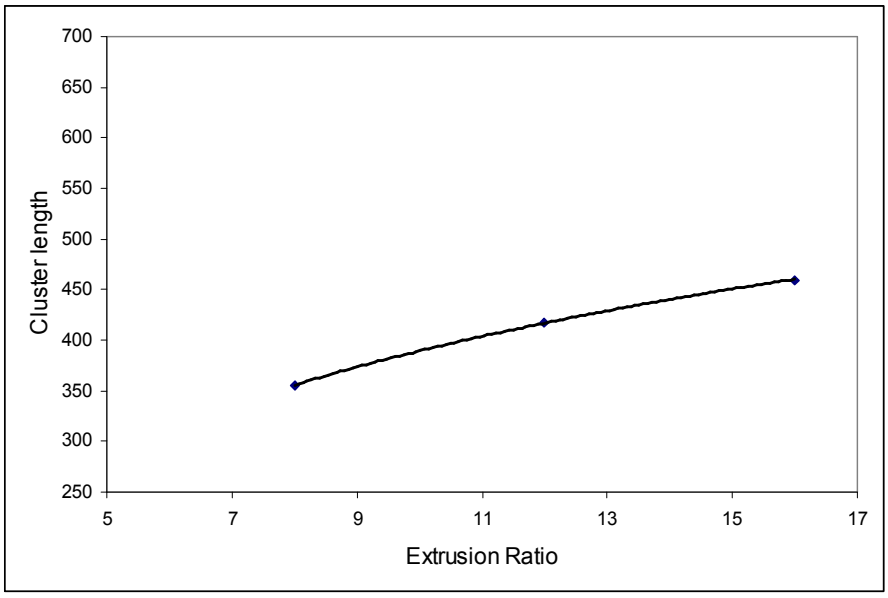



Figure 6.56: Relationship between extrusion ratio and cluster length at extrusion temperature 1010°C.

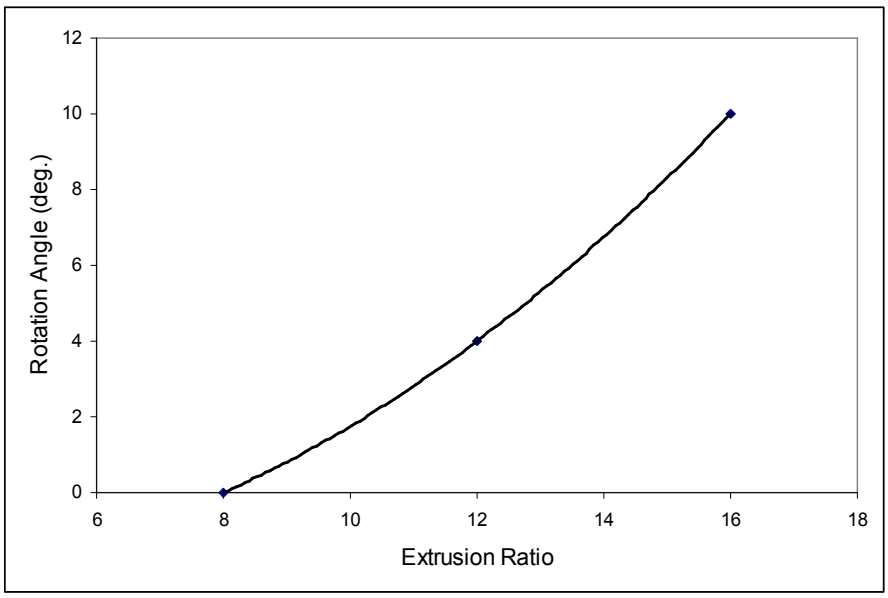



Figure 6.57: Relationship between extrusion ratio and rotation angle at extrusion temperature 1010°C.

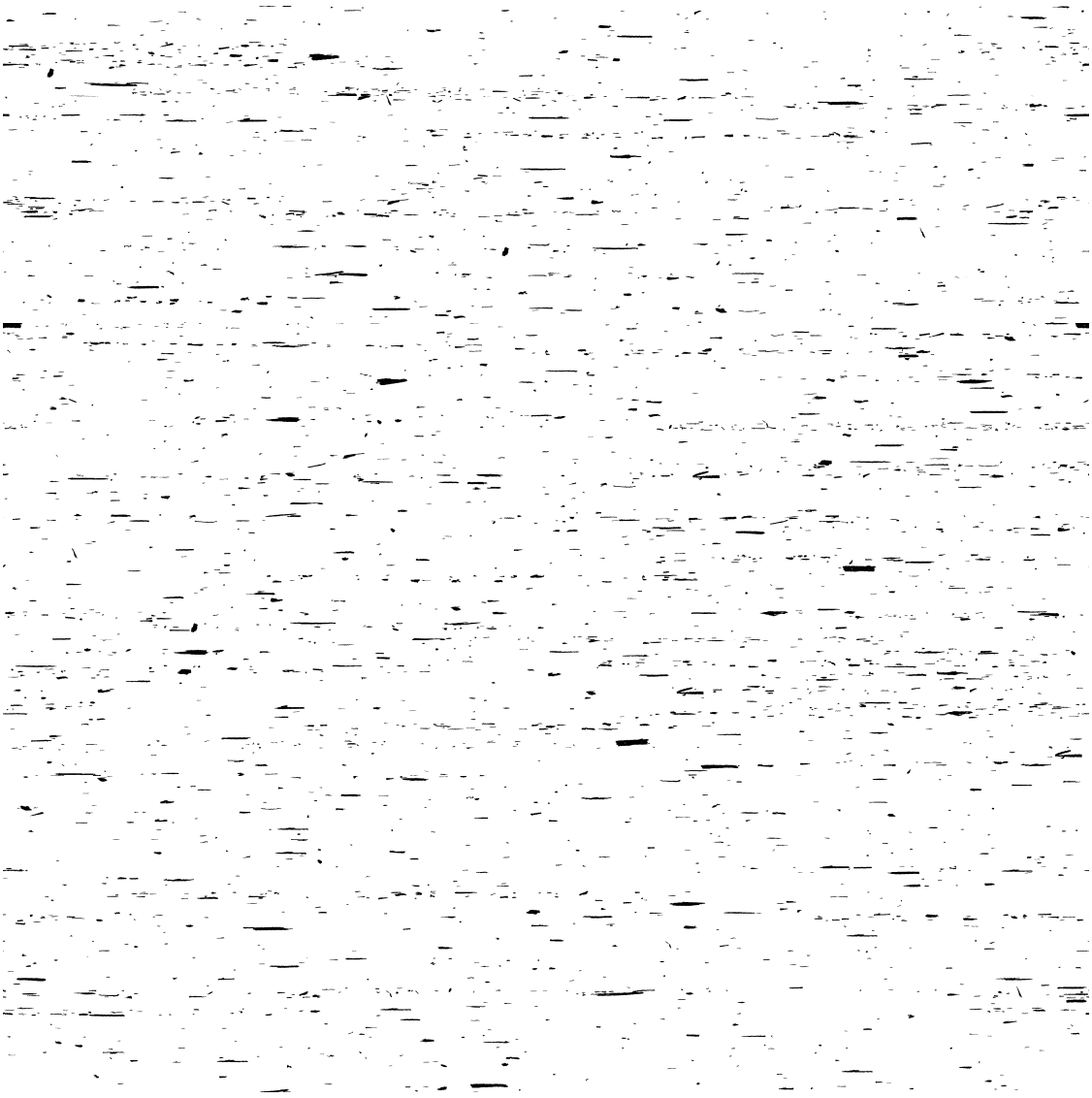


Figure 6.58: Virtual microstructure of boron modified Ti-6Al-4V alloy with extrusion temperature of 1010 °C and extrusion ratio of 14:1.

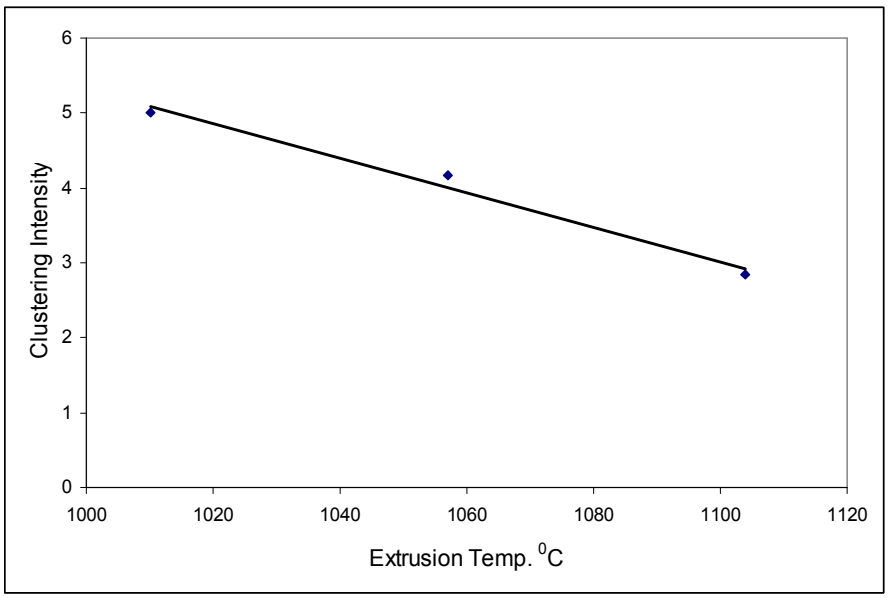



Figure 6.59: Relationship between extrusion temperature and clustering intensity.

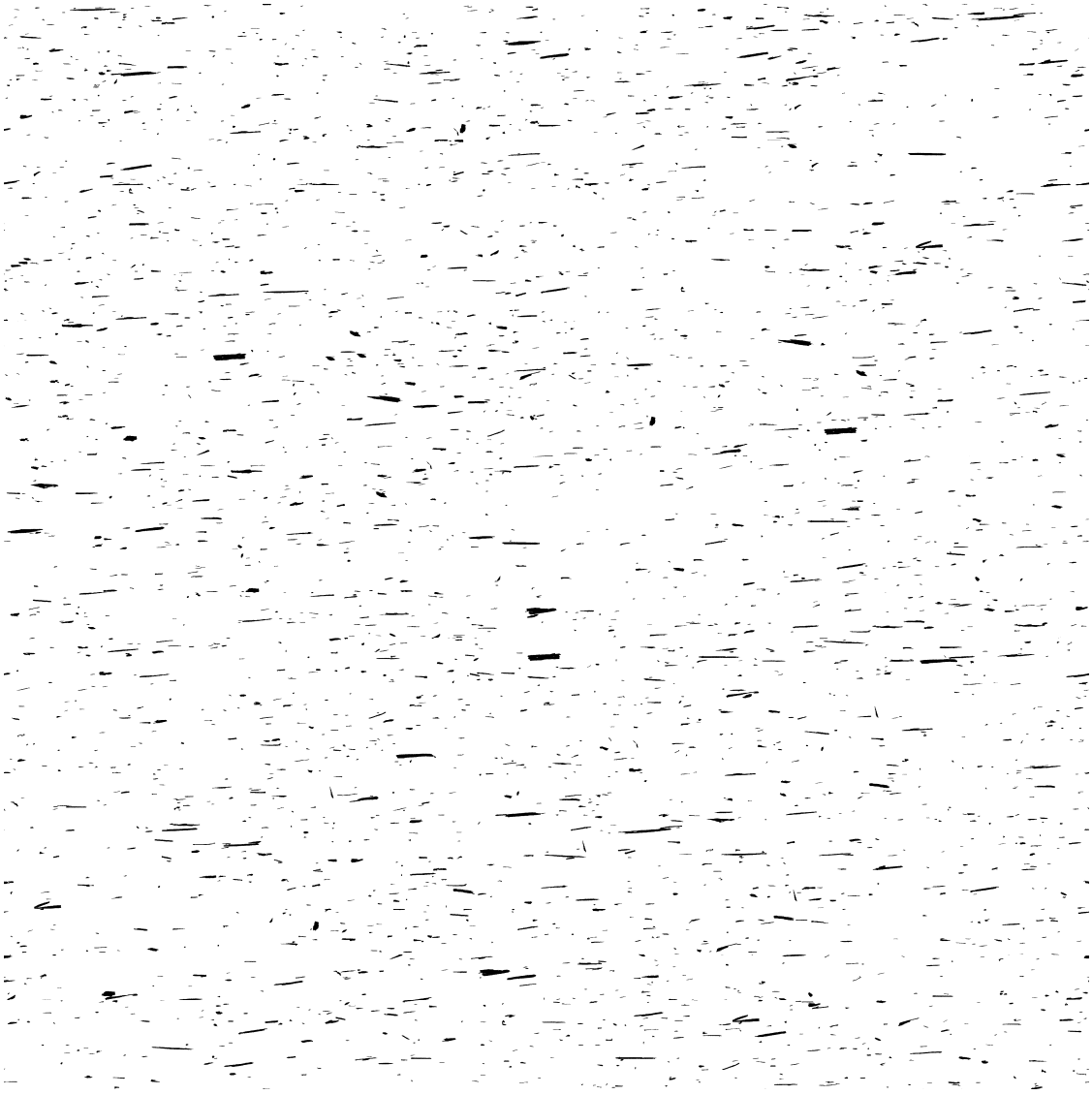


Figure 6.60: Virtual microstructure of boron modified Ti-6Al-4V alloy with extrusion temperature of 1080 °C and extrusion ratio of 8:1.

## 6.4 Recommendations for Future Research

The simulation-based materials design technique presented in this work describes the simulation of two-dimensional microstructures containing two phases. Similar methodology can be adopted to construct three-dimensional realistic and virtual

microstructures; the framework for the images and particle/morphologies can be expanded to include volumes instead of 2D surfaces, while the basic algorithm of generating the realistic microstructure and then utilizing the simulation parameters to create virtual microstructures can be applied in a similar manner as described in this research. Furthermore, the simulation model can itself be expanded to simulate microstructure having multi-phase and/or multi-scale constituents. Also, the techniques to estimate the mechanical behavior using 3D microstructures are still being researched. These different aspects of the simulation-based materials design methodology will all combine together to form a robust and flexible design technique that can be applied to a variety of material systems giving accurate predictions of material properties for given process parameters. The development for such a unified methodology would require several stages of collaborative research efforts.

# CHAPTER 7

# SUMMARY

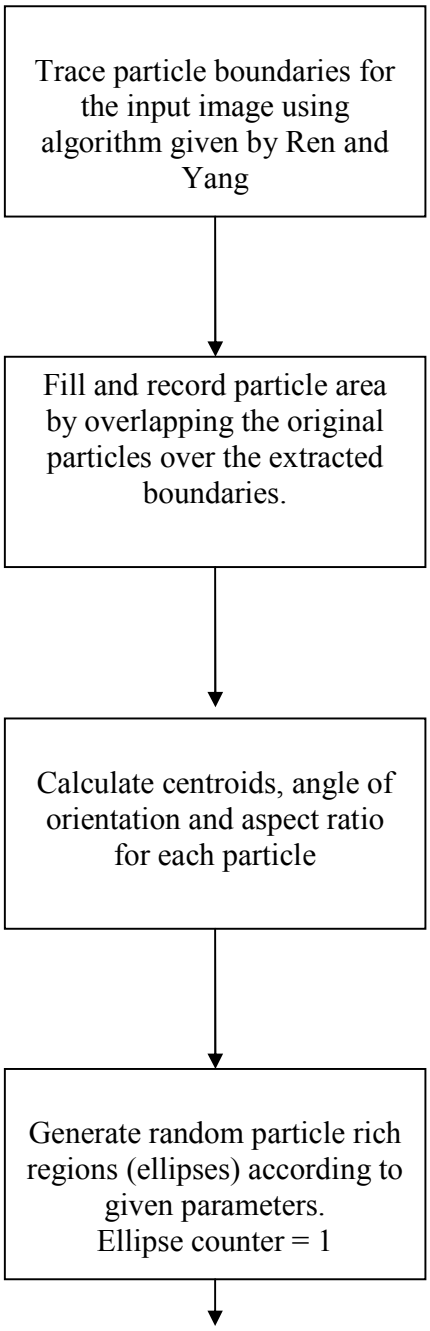
A novel simulation-based design methodology has been presented through this research. The key facets of this technology are to incorporate realistic/complex shapes/morphologies of the particle/features to generate large windows of realistic microstructures that are statistically similar to the corresponding real microstructures with respect to complex realistic particle/feature shapes/morphologies; spatial clustering and correlations; morphological orientation distributions; and size-shape-orientation distributions of the features. Statistical correlation functions, such as, two-point probability functions and lineal path distribution functions have been utilized to statistically compare the corresponding real and simulated microstructures. Once simulated microstructures are matched with the corresponding real microstructures, the simulation parameters are obtained. An important objective of this methodology is the creation of "virtual" microstructures by correlating the simulation parameters with the processing parameters for the given set of real microstructures and then interpolating/extrapolating the curves. These correlations can be utilized to generate a series of "virtual" microstructures representing a range of different processing parameters. The virtual microstructures created through this method do not require the physical manufacturing of the specimens and can be used to predict material properties using different computational models such finite-element analysis. This simulation-based design methodology results in a cost-effective and time-saving way of development of advanced materials. In the present work, the methodology has been presented via its application to the simulation of SiC particles in the DRA composites and TiB whiskers in

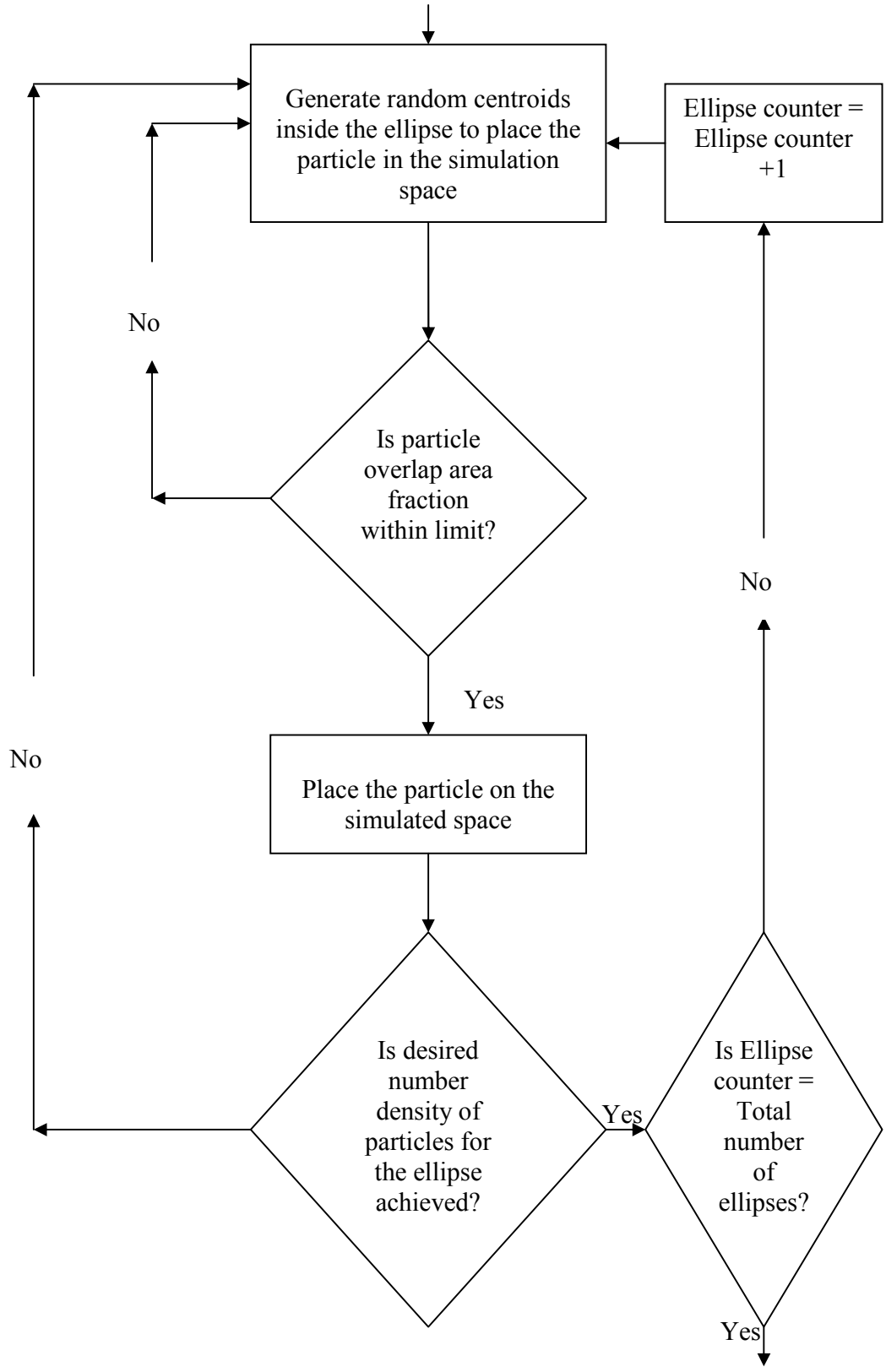
the boron modified Ti-alloys. The simulation of realistic microstructure those are statistically similar to the corresponding real microstructures of the materials under study have been presented. The simulation parameters used to create realistic microstructures of these materials have been correlated to the processing parameters and these relationships have been utilized to create virtual microstructure of the materials corresponding to various processing conditions other than those used to produce the real materials. These virtual microstructures represent a part of such an "atlas" of microstructures that can be generated via this technique.

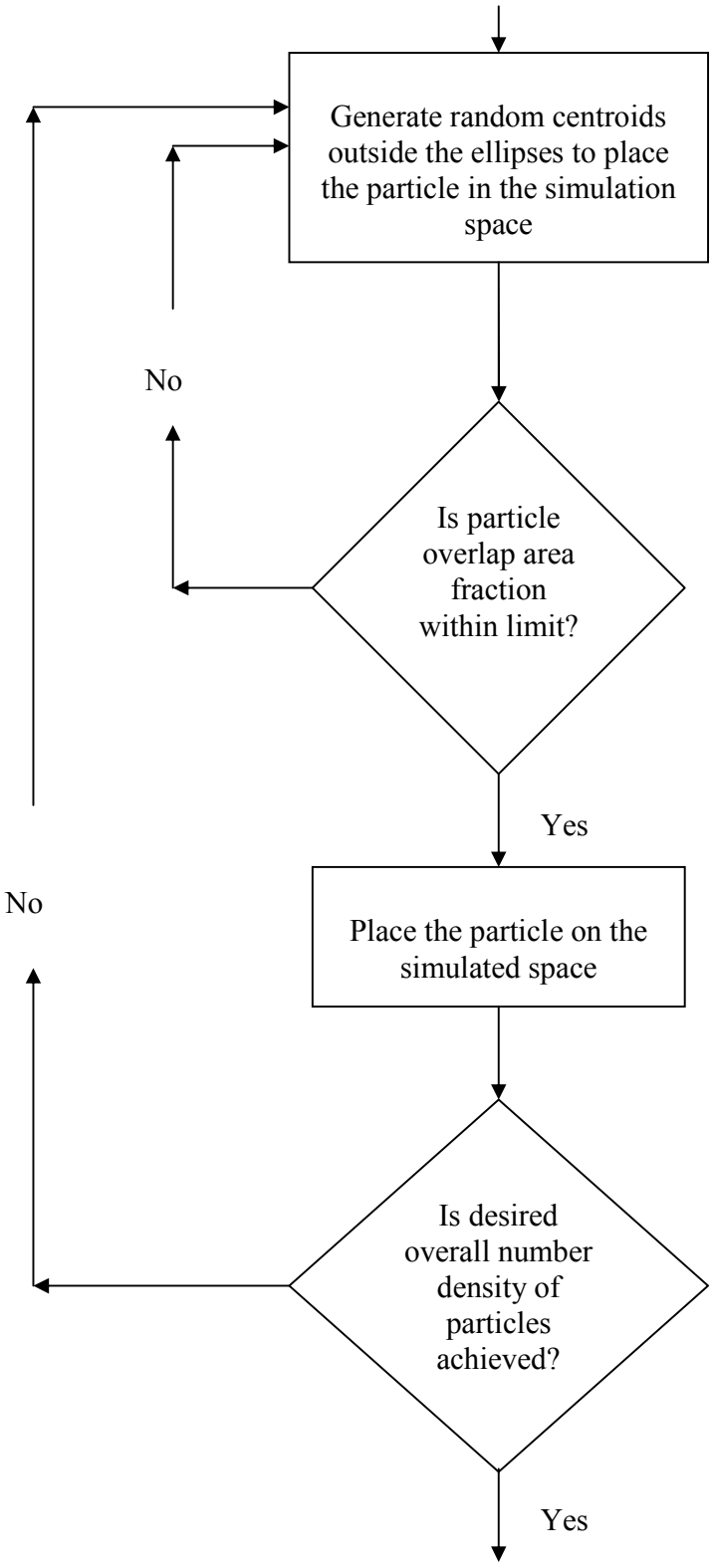
# APPENDIX A

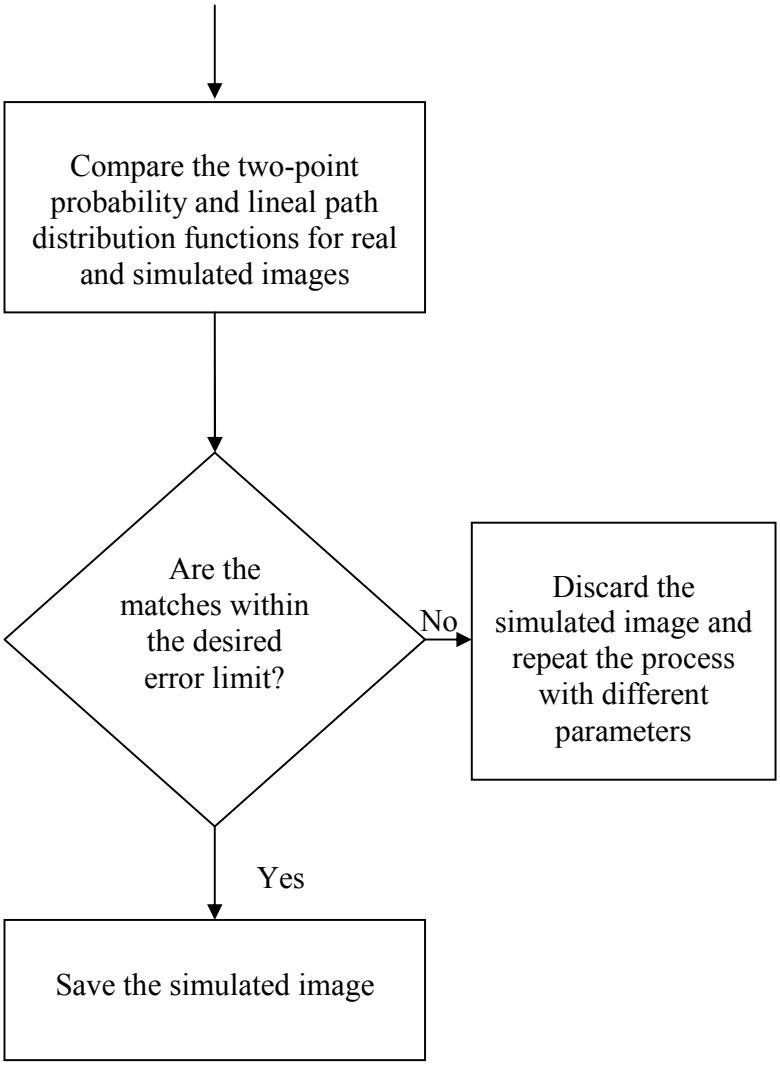
# SIMULATION FLOWCHART AND CODE

Simulation Flowchart




Generate random centroids inside the ellipse to place the particle in the simulation space
Ellipse counter = Ellipse counter +1
No
Is particle overlap area fraction within limit?
No
Yes
Place the particle on the simulated space
No
Is desired number density of particles for the ellipse achieved?
Yes
Is Ellipse counter = Total number of ellipses?
Yes


Generate random centroids outside the ellipses to place the particle in the simulation space
No
Is particle overlap area fraction within limit?
Yes
No
Place the particle on the simulated space
Is desired overall number density of particles achieved?
Yes


Compare the two-point probability and lineal path distribution functions for real and simulated images
Are the matches within the desired error limit?
No
Discard the simulated image and repeat the process with different parameters
Yes
Save the simulated image

Simulation Code

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <time.h>
#include <limits.h>
#include <math.h>
#include <iostream.h>

#define Pi 3.14159265358979
# define Im_in1(x,y) (*(buf_in1+((y-1)*x_s+x-1)))
# define Im_large_in(x,y) (*(large_in+((y-1)*(x_s+2)+x-1)))
# define Im_in2(x,y) (*(buf_in2+((y-1)*x_s+x-1)))
# define Im_out(x,y) *(buf_out+((y-1)*x_s+x-1))
# define Clip(x) ( (x) > (255) ? (255): (x))
# define Clip_min(x) ( (x) < (0) ? (0): (x))
# define min(x,y) ( (x)>(y)?(y):(x) )
# define max(x,y) ( (x)>(y)?(x):(y) )
#include "header.h"
#include <string.h>

int _stdcall C_Add_Sub_Mul(unsigned char Oper , unsigned char Opt, unsigned char
*buf_in1,unsigned char *buf_in2, unsigned char *buf_out, int x_s, int y_s, unsigned char
*debug_ar)
{

        unsigned char *large_in;

unsigned char *imgdata;
int *no_of_ellipse, nop,no_particles,rad;
int i,j,imageHeight,imageWidth,k;
float dist;
FILE *radius;


imageWidth=5000;
imageHeight=5000;  // width and height for the  simulated image


        imgdata=(unsigned char *) malloc (
(imageHeight+2)*(imageWidth+2)*sizeof(unsigned char));
```

```
        large_in =  ( unsigned char*)calloc( (x_s+2)*(y_s+2), sizeof( unsigned char ) );


        if(large_in == NULL ) return (666);
//
no_of_ellipse=&nop;
        struct particle *prtcl,*temp_bound, *sml_prtcl, *big_prtcl; //= new particle[1500];
        struct ellipse *ellp =new ellipse[200];




        prtcl=new particle[1300];
        temp_bound =new particle[1300];

        int nop_limit;
        nop_limit=1300;


        for (i=1;i<nop_limit;i++) {prtcl[i].area=0;    prtcl[i].perim=0;}

void sort( int arr[], int left, int right);

        FILE *fp,*fp1;

   struct chain
        {
                int x, y;
                struct chain *nextchain;
        };

   struct object1
        {
                struct chain  *chainhead;
                struct object1 *nextobject;
        };


        struct object1 *objhead, *objtemp, *objpass;
        struct chain *chaintemp, *chainpass;


         objhead=(struct object1 *)malloc(sizeof(struct object1));
```

```
    objpass=objhead=NULL;

            int fno,xoff,yoff; // xoff and yoff are variables to get the x and y offset for 1st
and last boundary points

    fno=1;
     char str[80], n1[10];
   itoa (fno,n1,10);
  strcpy (str,"4740-sim-");


          int offsetx[8]={1,1,0,-1,-1,-1,0,1};
          int offsety[8]={0,-1,-1,-1,0,1,1,1};

  // contour

          int prev_grey_val;
          int no_obj=0,m=0,l,tcode[7],is_obj=1,n=0;
          int xc, yc, xi, yi, ck,xobj,yobj,out1;

          int tcode0[8]={7,7,1,1,3,3,5,5};
          int k_out[8]={3,4,5,6,7,0,1,2};
          int k_inner[8]={7,0,1,2,3,4,5,6};
          int pcode, arr[500] ;
          float pixelum=1.0;
          int phi[7]={5,6,7,0,1,2,3};



          int *prt_info;
          prt_info= new  int[(x_s+2)*(y_s+2)];
          int prt_no,xx,yy;
          no_particles=0;

// to generate the boundary pixels of the particles in the input image
          bool bl; //4/9/07
for (i=0;i<x_s;i++)
                    for (j=0;j<y_s;j++)
                    prt_info[i+j*x_s]=0;
for (i = 0; i<=imageWidth; i++)
                                        for (j=0; j<=imageHeight;j++)
                    imgdata[i+j*imageWidth]=0;


                    for (i=1; i<=x_s;i++)
                    for (j=1; j<=y_s;j++)
```

```
                        Im_large_in(i+1,j+1) = Im_in1(i,j);//initial the value of
out_xy


                for (j=2; j<=y_s+1;j++)
                for (i=2; i<=x_s+1;i++)

                {
                        prev_grey_val = Im_large_in(i-1,j);
                        bl=Im_large_in(i,j)==255;

                    if ((Im_large_in(i,j)==255 && prev_grey_val==0) ||
(Im_large_in(i,j)==255 && prev_grey_val==2) )
                    {// This is outer-loop
                        Im_out(i-1,j-1)=255;


                        ////////////////////////////////
                        chainpass=(struct chain*)malloc(sizeof(struct chain));
//initial
                        xobj=xc = chainpass->x=i;
                        yobj=yc = chainpass->y=j;
                        chainpass->nextchain=NULL;
                      objtemp=(struct object1*)malloc(sizeof(struct object1));
                        objtemp->nextobject=NULL;
                        objtemp->chainhead=chainpass;

                        if(objhead==NULL) objhead=objtemp;
                        else objpass->nextobject=objtemp;
                        objpass=objtemp;

          ////////////////////////////////////////////

                        pcode = 7; // initial value at the start of the outer-loop

                        Im_large_in(i-1,j)=2;

                        l=0;

                        while (l<=6)
                        {

                          if (l==0)
                                tcode[l]=tcode0[pcode];
                          else
                                tcode[l]=(tcode[l-1]+1)%8;
```

```
                xi = xc + offsetx[tcode[l]];
            yi = yc + offsety[tcode[l]];

                if (Im_large_in(xi,yi)==0 || Im_large_in(xi,yi)==2)
                {
                        Im_large_in(xi,yi)=2;
                        l++;
                }
                else
                {
                        Im_out(xi-1,yi-1)=255;
                        Im_large_in(xi,yi)=3;
                        if (xi==xobj && yi==yobj)
                        {
                                ck=(tcode[l]+4)%8;
                                n=k_out[ck]+1;
                                xc=xi;
                                yc=yi;

                               if (n>6)
                                {
                                        l=n;
                                }
                               l=n;
                                while (n<=6)
                                        {
                                                tcode[l]=phi[n];
                                        tcode[l]=(ck+1)%8;
                                        xi = xc + offsetx[tcode[l]];
                                        yi = yc + offsety[tcode[l]];

                                        if (Im_large_in(xi,yi)==0 ||
Im_large_in(xi,yi)==2)
                                        {
                                                Im_large_in(xi,yi)=2;
                                                n++;
                                                l=n;
                                        }
                                        else
                                        {
                                                Im_large_in(xi,yi)=3;

                                                ////////////////////////////////
                                                        chaintemp=(struct
chain*)malloc(sizeof(struct chain));
```

```
                                                        chaintemp->x=xi;
                                                        chaintemp->y=yi;
                                                        chaintemp-
>nextchain=NULL;

                                                        chainpass-
>nextchain=chaintemp;
                                    chainpass=chaintemp;
                          ////////////////////////////////////////////
                                                            xc=xi;
                                                yc=yi;
                                                pcode=tcode[l];
                                                l=0;
                                                        n=7;

                                                   }


                                                }//end of n while

                                            }
                                            else
                                            {

                                                chaintemp=(struct
chain*)malloc(sizeof(struct chain));
                                                chaintemp->x=xi;
                                                chaintemp->y=yi;
                                                chaintemp->nextchain=NULL;

                                                chainpass->nextchain=chaintemp;
                                chainpass=chaintemp;
                            ////////////////////////////////////////////

                                                  xc=xi;
                                                  yc=yi;
                                                  pcode=tcode[l];
                                                  l=0;
                                            }

                                        }

                                    }//end of while

                no_obj++;
                                m=0;
```

```
                        } //end of the outer-loop if

                }// end of for i loop
                        fp=fopen ("output_contour.txt", "w");
                        fp1=fopen("output2.txt","w");
                objtemp=objhead;
                        l=1;
        while (objtemp != NULL)
        {
                chaintemp=objtemp->chainhead;
                k=1;
                prtcl[l].area=0;
                while (chaintemp != NULL)
                {
                //fprintf(fp, "%f %f\n", float (chaintemp->x*pixel),float (chaintemp-
>y*pixel));
                        fprintf(fp, "%6.4f %6.4f\n", chaintemp->x*pixelum-2, chaintemp-
>y*pixelum-2);
                        temp_bound[l].x[k]=(chaintemp->x*pixelum)-2;
                        temp_bound[l].y[k]=(chaintemp->y*pixelum)-2;

                        k++;
                chainpass=chaintemp->nextchain;
                chaintemp=chainpass;

                }

                temp_bound[l].perim=k-1;
                l++;
    fprintf(fp, "0000000 0000000\n");
    objpass=objtemp->nextobject;
          objtemp=objpass;
        }

        for (i=1;i<=l-1;i++){
                 for (j=1;j<=prtcl[i].perim;j++)
                        fprintf(fp1, "%d   %d\n", prtcl[i].x[j], prtcl[i].y[j]);

                        fprintf(fp1, "0000000 0000000\n");

                }



fprintf(fp, "1000000 %d\n", no_obj);fprintf(fp1, "1000000 %d\n", l-1);
fclose(fp);
```

```
////////////////////////////////
no_particles=l-1;
//// boundary capturing is complete


FILE *cut=fopen("cut.txt","w");
for (i=1;i<=no_particles;i++)
//for (j=1;j<=prtcl[i].area;j++)
fprintf (cut,"%d\t%d\n",(prtcl[i].x[prtcl[i].area]-prtcl[i].x[1]),(prtcl[i].y[prtcl[i].area]-
prtcl[i].y[1]));
fclose (cut);

// getting rid of inner boundary loops .....
l=0;
for (i=1;i<=no_particles;i++)
{


        xoff=temp_bound[i].x[temp_bound[i].perim]-temp_bound[i].x[1];
        yoff=temp_bound[i].y[temp_bound[i].perim]-temp_bound[i].y[1]; //difference
betweeen ending and the starting pixel on boundary

        out1=0;
        for (j=1;j<=temp_bound[i].perim;j++)
        {
                if (temp_bound[i].y[j]<temp_bound[i].y[1]) // if 'y' goes below the starting
point.. it must inner loop
                {

                        out1=1;
                        break;
                }

        }
        if (out1==1) continue;

        if ( (xoff==0) && (yoff==0))
        {
                for (j=1;j<=temp_bound[i].perim;j++)
                {
                        xx= temp_bound[i].x[j];
                        yy= temp_bound[i].y[j];

                        prt_info[xx+yy*x_s]=nop_limit+1; // putting marker on each
pixels
```

```
                }
                continue;

        }


//      if ((yoff==1) || ( (yoff==0)&&(xoff==1) )) //dont really need this now 6/1/07
//      {
                l++;
                if (l==258)
                        l=258; //
                prtcl[l].area=0;
                for (j=1;j<=temp_bound[i].perim;j++)
                {

                        k=j;
                        xx=temp_bound[i].x[j];
                        yy=temp_bound[i].y[j];



                        if (prt_info[xx+yy*x_s]==0)                     { // just to make sure
that no boundary pixel is counted twice .. which may happen in a single pixel line
                        prt_info[xx+yy*x_s]=l; // putting the particle no. on the boundaries
..on the prt_info plane...

                        prtcl[l].perim++;
                        prtcl[l].x[prtcl[l].perim]=temp_bound[i].x[j];
                        prtcl[l].y[prtcl[l].perim]=temp_bound[i].y[j];


                        prtcl[l].area++;
                        prtcl[l].fillx[prtcl[l].area]=xx;
                        prtcl[l].filly[prtcl[l].area]=yy;


                        }


                }



}
no_particles=l;
// only outer loops remain .....
```

```
for (i=0;i<x_s;i++)
		for (j=0;j<y_s;j++)
		{
			if ((i==88) && (j==26))
				{
					i=88;
				}

			if ( (prt_info[i+j*x_s]==0)&& (Im_in1(i+1,j+1)==255) )  //
Im_in1(i+1,j+1) reads the same pixel as prtinfo[i+j*x_s]
			{
				for(k=i-1;k<=i+1;k++)
				{
					for(l=j-1;l<=j+1;l++)
					{
						out1=0;

						if (prt_info[k+l*x_s]>0)
						{
							prt_no=prt_info[k+l*x_s];



							prt_info[i+j*x_s]=prt_no;
							prtcl[prt_no].area++;

	prtcl[prt_no].fillx[prtcl[prt_no].area]=i;

	prtcl[prt_no].filly[prtcl[prt_no].area]=j;

							out1=1;
							break;
						}

					}
					if (out1==1) break;
				}
			}
		}


		delete [] temp_bound;

		temp_bound=new  particle[nop_limit];
		l=0;
```

```
		for (i=1;i<=no_particles;i++)
		{
			out1=0;
			for (j=1;j<=prtcl[i].perim;j++)
			{
				if ((prtcl[i].x[j]==0) || (prtcl[i].y[j]==0) ||
(prtcl[i].y[j]==y_s-1) || (prtcl[i].x[j]==x_s-1))
				{
					out1=1;
					break;
				}

			}
			if (out1==1) continue;
			l++;
			temp_bound[l]=prtcl[i];
		}
		no_particles=l;
		delete [] prtcl;

		prtcl=new particle[nop_limit];
		for(i=1;i<=no_particles;i++)
			prtcl[i]=temp_bound[i];

		xc_yc_area( prtcl,   no_particles,x_s,y_s);  // To calculate the area and Xc
and Yc for each particle

		int radi;
fclose(fp1);




float *rst;
const MM=5;
float result[MM+1];
rst=&result[0];


simulate(ellp,prtcl,imgdata,imageWidth,imageHeight,no_of_ellipse,no_particles,rst); //
//To generate the simulation

strcat (str,"-test.bmp");
```

```
        for (i=0;i<=imageHeight;i++)
                for(j=0;j<=imageWidth;j++)
                        imgdata[j+i*imageWidth]=0;
        i=1;
        j=1;


        delete [] ellp;



return (101);
}

void sort( int arr[], int left, int right)
{
        int i,last;
        void swap(int arr[],int i ,int j);
        if (left>=right)
                return;
        swap(arr,left,(left+right)/2);
        last =left;
        for (i=left+1; i<=right;i++)
                if (arr[i]<arr[left])
                        swap(arr, ++last,i);
                swap(arr,left,last);
                sort(arr,left,last-1);
                sort(arr,left+1,right);
}
void swap(int arr[],int i,int j)
{
        int temp;

        temp =arr[i];
        arr[i]=arr[j];
        arr[j]=temp;
}
```

// To calculate the centroids for each particle

```
struct particle {
		int x[1000];
		int	y[1000];   //boundary pixels
		int area;
		int perim;
		int xc;
		int yc;
		int fillx[9000];
		int filly[9000];   // area pixel of each particle
		float asp_ratio;  //aspect ratio
		float angle;	  //angle
	};
 void xc_yc_area(struct particle *prtcl, int no_particles,int x_s,int y_s)
{


	struct point
	{
		int xx; int yy;
	};

	struct point *pnt1, *pnt2;
	unsigned char *temp_img;
	temp_img = (unsigned char*) calloc ( (x_s+2)*(y_s+2), sizeof (unsigned char));

	pnt1=(struct point*)malloc( 50000* sizeof(struct point));
	pnt2=(struct point*)malloc( 50000* sizeof(struct point));
	FILE *fp=fopen("check7.txt","w");
	void sort( int array[], int left, int right);
	int i,j,ip,count,yp,yp2,array[5000],xp,jp,p1,p2,xp2,min,max;
	int x2[3000],y2[3000],x[3000],y[3000],size;
	int all_p,up,tot_len,xc,yc,length,sum;
	for (i=1;i<=no_particles;i++)
		for(j=1;j<=prtcl[i].area;j++)
			temp_img[prtcl[i].fillx[j]+(prtcl[i].filly[j])*x_s]=255;

	for (all_p=1;all_p<=no_particles;all_p++)
	{
		if (all_p==32)
			all_p=32;

		tot_len=0;
		up=0;
		size=prtcl[all_p].perim;
		for (i=1;i<=size;i++) {x[i]=prtcl[all_p].x[i];y[i]=prtcl[all_p].y[i];}
```

```
for (i=1;i<=size;i++) {x2[i]=x[i];y2[i]=y[i];}
p1=1;p2=1;


for (ip=1;ip<=size;ip++)
{

        for (i=1;i<=500;i++)
                array[i]=0;
        count=1;
        yp=y[ip];//bonds(ip,2);
        array[count]=x[ip];//bonds(ip,1);
   if (yp==20000)  continue;



        for (jp=ip+1;jp<=size;jp++)
        {   //bonds22{nop,2}
                yp2=y[jp];//bonds(jp,2);
                if (all_p==401 && ip==70)
                {
                        ip=70;
                }
                if (yp2==20000) continue;

          if (yp2==yp)
                {
              count=count+1;
        array[count]=x[jp];//bonds(jp,1); % putting X co-ordinate in an
array.. for same Ys
                        y[jp]=20000;//bonds(jp,2)=0;
                }
        }

         min=array[1];
         max=array[1];
        for(i=1;i<=count;i++)
        {

                if (array[i]<min) min=array[i];
                if (array[i]>max) max=array[i];
        }
        xp=min;

        if (count==1)
```

```
        {
                xc=array[count];
                length=1;
                sum=xc;
        }
        else
        {
                sum=0;length=0;
                for(i=min;i<=max;i++)
                {
                        if (temp_img[i+yp*x_s]==255)
                        {
                                sum=sum+i;
                                length++;
                        }
                }



        }

        tot_len=tot_len+length;
        up=up+sum;
}
if (tot_len==0) tot_len=1;
prtcl[all_p].xc=up/tot_len;
tot_len=0;
up=0;




for (i=1;i<=size;i++) {x[i]=x2[i];y[i]=y2[i];}
for (ip=1;ip<=size;ip++)
{       //to calculate yc and area for an particle.
        for (i=1;i<=500;i++) array[i]=0;
   count=1;
   xp=x[ip];
   array[count]=y[ip];
   if (xp==20000)  continue;



        for (jp=ip+1;jp<=size;jp++)
        {
```

```
                xp2=x[jp];
                if (xp2==20000) continue;

                if (xp2==xp)
                {
                        count=count+1;
                        array[count]=y[jp];// putting Y co-ordinate in an
array.. for same Xs
                        x[jp]=20000;
                }
        }

        fprintf(fp,"%d\n",count);

         min=array[1];
         max=array[1];
        for(i=1;i<=count;i++)
        {

                if (array[i]<min) min=array[i];
                if (array[i]>max) max=array[i];
        }


        yp=min;

        if (count==1)
        {
                 yc=array[count];
                 length=1;
                 sum=yc;
        }
        else
        {
                sum=0;length=0;
                for(i=min;i<=max;i++)
                {
                        if (temp_img[xp+i*x_s]==255)
                        {
                                sum=sum+i;
                                length++;
                        }
                }


        }
```

```
//                          if (count>1)
//                             length=length+1; //% correction ..to count both end points of
line in the 'length'

                            tot_len=tot_len+length;
                            up=up+sum;//yc*length;
                    }
                    if (tot_len==0) tot_len=1;
                    prtcl[all_p].yc=up/tot_len;

          }//end for  'all_p' where all_p goes for 1 to all particles.
          fclose(fp);
}

#include <time.h>
#include <stdio.h>
#include <limits.h>
#include <math.h>
#include <iostream.h>
#include <windows.h>
#include <fstream.h>


#define Pi 3.14159265358979

#define in_one_cell 20

#include "Rand_fast_ulong.h"
#include "Rand_53bit.h"
#include "2D_Class.h"
#include "3D_Class.h"
#include "erf.h"
#include <string.h>


struct particle {
                    int x[1000];
                    int       y[1000];
                    int area;
                    int perim;
                    int xc;
                    int yc;
                    int fillx[9000];
                    int filly[9000];
```

```
		float asp_ratio;
		float angle;
	};

void check(struct ellipse *ellp )
{

	ellp[1].x=10;

}

// check for the image .... it reads the image upside down.....!!!!!
unsigned char *LoadBitmapFile(char *filename, BITMAPINFOHEADER
*bitmapInfoHeader)
{
	FILE *filePtr; //our file pointer
	BITMAPFILEHEADER bitmapFileHeader; //our bitmap file header
	unsigned char *bitmapImage; //store image data
	int imageIdx=0; //image index counter
	unsigned char tempRGB; //our swap variable

	//open filename in read binary mode
	filePtr = fopen(filename,"rb");
	if (filePtr == NULL)
	{	printf(" where is the file ?\n ");
		return NULL;
	}

	//read the bitmap file header
	fread(&bitmapFileHeader, sizeof(BITMAPFILEHEADER),1,filePtr);

	//verify that this is a bmp file by check bitmap id
	if (bitmapFileHeader.bfType !=0x4D42)
	{
		fclose(filePtr);
		return NULL;
	}

	//read the bitmap info header
	fread(bitmapInfoHeader, sizeof(BITMAPINFOHEADER),1,filePtr);

	//move file point to the begging of bitmap data
	fseek(filePtr, bitmapFileHeader.bfOffBits, SEEK_SET);

	//allocate enough memory for the bitmap image data
	bitmapImage = (unsigned char*)malloc(bitmapInfoHeader->biSizeImage);
```

```
	//verify memory allocation
	if (!bitmapImage)
	{
		free(bitmapImage);
		fclose(filePtr);
		return NULL;
	}

	//read in the bitmap image data
	fread(bitmapImage,bitmapInfoHeader->biSizeImage,1,filePtr);

	//make sure bitmap image data was read
	if (bitmapImage == NULL)
	{
		fclose(filePtr);
		return NULL;
	}



		//close file and return bitmap iamge data
		fclose(filePtr);
		return bitmapImage;
}

void simulate(struct ellipse *ellp ,struct particle *prtcl,unsigned char *imgdata,int imageWidth,int imageHeight,int *noe,int original_nop, float *result)

 {
	long SimID;
   long Ntot, Ptotout, Ptotin;// Lx, Ly;
	double Aa, Area, Na, PAain, PAaout;
	int Lx,Ly;
	Lx=imageWidth;Ly=imageHeight;
	void sort1( float arr[], int arr2[],int left, int right);
	void sort ( int arr[],int left, int right);
	unsigned char *temp_layer;
	temp_layer=(unsigned char *) malloc ( (imageHeight+2)*(imageWidth+2)*sizeof(unsigned char));
	int prtcl_window;

	struct elipse_size
	{
	//double x,y,r;
	double elipa,elipb;
```

```
};

struct part
{
double x, y, r;
};
int fno;
float    lower_limit;
fno=0;



struct errr
{
        int fn; // file no.
        float mxerr; // max error
        float agerr; // avg error
};
errr err[11],erry[11],err45[11];

struct parameters
{
        int x;
        int y;
        float Aa;
        float lwr_lmt;
};
parameters pmt[800];


part* outside_array;
part* inside_array;

elipse_size* size_array;
double mean_sizea, mean_sizeb;

char *fname1, *fname2;
ellp[1].x=7;
long i,j,w; //loop variables
double ex,ey,ea,eb;

struct point
{
        int x,y;
};
struct big_particle_data_2  //asp_rat, alpha, T.x, T.y
```

```
	{
		float asp_ratio,angle;
		int x,y; // these are the co-ordinates of the centroid of the particle on the full image. T.x and T.y

	};
	struct big_particle_data_2 *big_part_data_2;
	struct pixel_particle_info
	{
		int xc;  // x center for the particle
		int yc;   // y center for the particle
		int ip;    // particle identification no. ( particle no. in the array)
		float ovrlpd_pixels;
	};
	struct particle_info // this is for all the particles in the image
	{
		int xc;  // x center for the particle
		int yc;   // y center for the particle
		int ip;    // particle identification no. ( particle no. in the array)
		float angle; //angle at which the particle is placed
//		float init_angle; // angle at which the particle is placed before ne rotation ( angle after the simulation)
	};
	struct overlapped_particles
	{
		int xc;
		int yc;
		float prt_area;		// area of particle under consideration
		float ovrlpd_pixels; // no. of pixels being overlapped
	};
	struct particle_info *part_info;
	struct point F1,F2,T;
	struct pixel_particle_info *px_prt_info;
	struct overlapped_particles *ovrlpd_prt;
	int overlap_check;

	float	overlap_ratio;//=0.292;
	 //	VOLUME FRACTION FOR THE OUTPUT

	char str3[80];


	Area=Lx*Ly;
	FILE *errfile;
```

```
//        fname1="mono.txt";
          FILE *stream, *stream2,*stream3,*fp1;
          float  ptreal[501],ptrealy[501],ptreal45[501],error[501],avgerr,maxerr,toterr;
    float lpreal [501];

          stream =fopen ("lineal-path-real-x-E.txt","rt");
          int siza[4], sizb[4];
          double freq[4], freq_sum, narea=0;
          freq_sum=0;narea=1;

          for (i=0;i<=500;i++)
          {

                    fscanf( stream, "%f ",&ptreal[i]);

                    lpreal[i]=ptreal[i];

          }
          fclose(stream);


/
          for (i = 1; i<=10;i++)
          {
                    err[i].mxerr=200;erry[i].mxerr=200;err45[i].mxerr=200;

          }

                    int writeGrayScaleDataToBmpFile(char * , unsigned int   , unsigned int ,
unsigned char * );

BITMAPINFOHEADER bitmapInfoHeader;
unsigned char *bitmapData;
float rot_angle (unsigned char * , unsigned char * , float ,int ,int , int , int ,int , int ,
int,int,int, int,float ) ;

float asp_rat( unsigned char *, int, int);
int borderx ;
int bordery ;

float frame_area ;
int        af ,sum_area_in ,tot_area_out ;

                    char *imgname ;
          int ellipse_boundary  ;
```

```
int       particle_outside,l ;
int       particle_inside ;
int       points ,xp,yp,remove,out;
int       xtrap ,current_size=original_nop;
float     tot_area ;
float     upper_limit ;



FILE *lpath;
char lpath_name[80];
int start,end,unit_r,unit_xy,frame_len,cnt,len_cnt,counter;
float ovrlp_counter;
int out1,new_prt;
float in_pixel_count,overall_pixel_count,temp_in_pixel_cnt,temp_overall_pixel_cnt;
const MM=500;
char str[80], n1[10],str2[80];
          unsigned long p[4][MM],q[MM];
          float res[MM];

          int  pd;
          int xa,ya,xb,yb,r,x_s,y_s,pa,pb;
          double temp_sizea, temp_sizeb;
          long ran_n;double cell_size_x, cell_size_y, ratioab;double rac,rbc,xc,yc;
                    int k,no_of_ellipse;
          int no_ovrlpd_prt;
          FILE *part_info_file;
int dist_x,dist_y;
int ip ,cp;float area_elp,f_length;
int np ;long xx,yy;
int done,lft,rht,up,dwn,          particle_counter;
int check ,check1,tot_no_part ; // tot_no_part counts the total no. of particles in the final
image... .

int finish,worstr; // its the r value where maxerr takes place
FILE *out123,*fp, *ch  , *ch9 ,*ch4, *outcheck, *big_particle_file, *angle_file ; //*ch8 ,
*fp2,*ch10
long lx,ly; // lx, ly are the no. of cells in x and y directions
float     tot_elp_area      ,d1,d2,area_frac,overall_ar_frac,sum_in_ar_fr      ,inside_ar_frac,
out_ar_frac ;
double randx,randy;
int counter_check[1000],only_points=0;
int pty,ang,type_of_ellp, no_sims;
int tst;
float vol_frac;
float alpha_max=0;
```

```
float alpha_rot=3; // the angle by which the particles are rotated (after making them rotate randamly upto alpha_max)
float clust_inten=4;    // for complete random orientations alpha_rot=0    ,    for    no rotation its 90
                                                    // maximum angle which will remain after alpha_rot would be (alpha_max/2)-alpha_rot
char imagename[30];

int orig_prt_area; // this is the area of particle before rotation....
start=1;end=150;unit_r=1;unit_xy=1;
overlap_check=1;overlap_ratio=0;       // OVERLAP FRACTION
ang=501;
prtcl_window=500;
float aspect_ratio,avg_size,avg_aspect_ratio,avg_angle,temp_angle;
int big_particle_number=0; //
int angle_bin_arr[40][4],prt_size_bin[40][4],no_of_bin;


        for (i=0;i<=imageWidth;i++)
        for (j=0;j<=imageHeight;j++)
                temp_layer[i+j*imageWidth]=0;
type_of_ellp=1;freq[1]=1;freq[2]=3;
        errfile=fopen("error-file.txt","w");
        fprintf(errfile,"img\t");
        for ( i =1;i<=type_of_ellp;i++)
                fprintf(errfile,"x\ty\tfreq\t");
        fprintf(errfile,"Aa\tlwr limit\tavg. err.\tmax. err\n");
        fclose(errfile);

        int big_part_no1,big_part_no2,big_part_no3,big_part_data[10][5];

        if (big_particle_number>0){
        big_particle_file=fopen("big_particle_data.txt","r");
        for (i=1;i<=18;i++)
                for (j=1;j<=4;j++)
                        fscanf (big_particle_file,"%d",&big_part_data[i][j]);

                fclose(big_particle_file); } //*/
        int t;
        float radius,theta,alpha; // these variables are used for rotation of particles...
        t=alpha_max-2*alpha_rot;// this is to be used for the naming of the file
        alpha=alpha_max*Pi/180;alpha_rot=alpha_rot*Pi/180;
        float random_no1[20];int random_no2[20],big_particles;


        char big_part_name[25];
```

```
        unsigned char *big_part_img, *small_part_img;
        int big_part_xc=358,big_part_yc=94,big_part_area=40151;

        big_part_img=(unsigned char *) malloc ( (683)*(683)*sizeof(unsigned char));
        small_part_img=(unsigned char *) malloc ( (500)*(120)*sizeof(unsigned char));
// this is to store small particles
        for(i=0;i<683;i++)
                 for (j=0;j<683;j++)
                          big_part_img[i+j*683]=0;
        siza[2]=250;sizb[2]=250;no_sims=1;  // 9/7

        strcpy (imagename,"TiB_DOE_E");vol_frac=.014493;//Aa was .15 //      0.0315
        lower_limit=vol_frac*clust_inten;Aa=0.14;sizb[1]=40;siza[1]=1750; //it was .459
...150 and 1750

        ip=1;

        /// ASPECT RATIO AND PRT SIZE CALCULATIONS AND FILE READING -
----------------------------------------

                FILE *aspect_ratio_file;

                FILE *prt_size_file;
                                prt_size_file=fopen("prt_size_file-A-4.txt","w"); // to print
out the prt_area of input particles.
                                for(i=1;i<=original_nop;i++) {fprintf (prt_size_file,"%d
        %d      %d      %d      %d\n",i,prtcl[i].xc,prtcl[i].yc,prtcl[i].area,prtcl[i].perim);}

                                fclose(prt_size_file); //*/

                         i=3;

// to calculate the aspect ratio

                aspect_ratio_file=fopen("aspect_ratio.txt","w");
                for(i=1;i<=original_nop;i++)
                        {
                                        for(k=0;k<300;k++)
                                                for (l=0;l<120;l++)
                                                        small_part_img[k+l*300]=0;
                                        if              (i==379)                        {//
writeGrayScaleDataToBmpFile("part136.bmp", 300, 120, small_part_img);
                                                i=379;}// particle transfer from 'prtcl' to
small_part_data ....
                                        for (k=1;k<=prtcl[i].area;k++)
                                        {
```

```
                //                                      xx=prtcl[ip].fillx[i]-
prtcl[ip].xc;//+T.x;
                //                                      yy=prtcl[ip].filly[i]-
prtcl[ip].yc;//+T.y;
                                                        xx=prtcl[i].fillx[k]-
prtcl[i].xc+150;
                                                        yy=prtcl[i].filly[k]-
prtcl[i].yc+60;

        small_part_img[xx+yy*300]=255;
                                        }
                                //
        writeGrayScaleDataToBmpFile("temp.bmp",300,120,small_part_img);
                                        aspect_ratio=asp_rat(small_part_img,300,120);
                                        fprintf (aspect_ratio_file,"%f\n",aspect_ratio);
//                                      prtcl[i].asp_ratio=aspect_ratio;

                        }
                        fclose(aspect_ratio_file);   */



                                for(k=0;k<300;k++)
                                        for (l=0;l<120;l++)
                                                small_part_img[k+l*300]=0;



                        aspect_ratio_file=fopen("aspect_ratio.txt","r");
                        for(i=1;i<=original_nop;i++)
                                fscanf(aspect_ratio_file,"%f",&prtcl[i].asp_ratio);
                        fclose(aspect_ratio_file);
                        aspect_ratio_file=fopen("ferret_angles.txt","r");     // asp_rat and
fer_ang are for the input sample ( which'll be the same for all the simulations)
                        for(i=1;i<=original_nop;i++)
                                fscanf(aspect_ratio_file,"%f",&prtcl[i].angle);
                        fclose(aspect_ratio_file); // just used the same FILE variable for
ferret_angles too
                        for(i=1;i<=original_nop;i++)
                                prtcl[i].angle=-1*prtcl[i].angle; // the axis for calculations
of angles was opposite to placing the particles!

                        aspect_ratio_file=fopen("prt_size_dist-doe-E.txt","r"); // this is for
the real sample .. not the input file
                        i=0;
                        while (!feof(aspect_ratio_file))
```

```
                        {
                                i++;
                                fscanf(aspect_ratio_file,"%d",&prt_size_bin[i][0]);
                                fscanf(aspect_ratio_file,"%d",&prt_size_bin[i][1]);
                        }
                        no_of_bin=i;
                        fclose(aspect_ratio_file);
                        for(i=1;i<=no_of_bin;i++)
                                prt_size_bin[i][2]=prt_size_bin[i][1]+floor
((prt_size_bin[i][1]*3/100)+.5); //percentage for particle size distribution here


                        avg_size=0.0;avg_aspect_ratio=0.0;avg_angle=0.0;
                        for(i=1;i<=original_nop;i++)
                        {
                                avg_size=avg_size+prtcl[i].area;
                                avg_aspect_ratio=avg_aspect_ratio+prtcl[i].asp_ratio;
                                avg_angle=avg_angle+fabs(prtcl[i].angle);
                        }
                        avg_size=avg_size/original_nop;
                        avg_aspect_ratio=avg_aspect_ratio/original_nop;
                        avg_angle=avg_angle/original_nop;
                        i=3;

// ASPECT RATIO calculated


        float theta_rot; // the new angle at which the particle must be placed.. (before
checking for the overlap)
        int direction,rot_counter; // to get the sign of the angle, to count the iterations of
rotations applied on the img_data

lower_limit=vol_frac*clust_inten;


        while (fno<no_sims){
        fno++;

                for(i=1;i<=no_of_bin;i++)
                                prt_size_bin[i][3]=0; // initializing the prt_size _bin for the
output image .. before each simulation
        tst=0;
        px_prt_info=  new    pixel_particle_info[(imageWidth+2)*(imageHeight+2)];//////
10/3
        ovrlpd_prt=new  overlapped_particles[15];//// 10/3
```

```
	big_part_data_2= new big_particle_data_2[10];
	part_info= new particle_info[7000];
	part_info_file=fopen("part_info_file.txt","w");
	fprintf(part_info_file,"out prt no\tout Xc\tout Yc\tInp prt no\tin Xc\tin Yc\n");

		SimID = (unsigned)time( NULL );
	seedMT(SimID);

	for (i=1;i<=18;i++)
	{
		random_no1[i]=random53();
		random_no2[i]=i;
	}

	sort1 (random_no1,random_no2,1,18);



	randx=random53();big_part_no1=1+3*randx;

	int big_particles_counter;



	 randx=random53();
	 T.x=360+randx*4300;// this is to generate xc for the simulated image
	 randy=random53();
	 T.y=360+randy*4300;


	 overall_pixel_count=0.0;
	 for (
big_particles_counter=1;big_particles_counter<=big_particle_number;big_particles_cou
nter++) //
	 {
		big_particles=big_particles_counter;
/*		if (big_particles>3)
			big_particles=big_particles-3;*/
		//big_part_no1=random_no2[big_particles];
		alpha=(alpha_max*random53())*Pi/180;
		alpha=alpha-(alpha_max*Pi/180)/2;// 7/3/06 to make the rotation angle
between -alpha_max/2 to +alpha_max/2 instead of 0 to alpha_max
//		if (fabs(alpha)<alpha_rot)
//			alpha =0;
//		else
```

```
//                      alpha=alpha-(alpha/fabs(alpha))*alpha_rot; // before this add the step where the angle of rotation is adjusted

        // for the size, the angle of orientation and aspect ratio.

                big_part_no1=big_particles;
        //       alpha = 0;

                 itoa (big_part_no1,n1,10);
                strcpy  (big_part_name,"input_big_part-");
                strcat  (big_part_name,n1);
                strcat  (big_part_name,".bmp");


                big_part_img=LoadBitmapFile(big_part_name,&bitmapInfoHeader);

writeGrayScaleDataToBmpFile("temp.bmp",big_part_data[big_part_no1][1],big_part_data[big_part_no1][2],big_part_img);
                 big_part_img=LoadBitmapFile("temp.bmp",&bitmapInfoHeader);


        aspect_ratio=asp_rat(big_part_img,big_part_data[big_part_no1][1],big_part_data[big_part_no1][2]);
                big_part_data_2[big_part_no1].asp_ratio=aspect_ratio;

                        finish=0;
                        while (finish==0)
                        {
                                finish=1;
                                randx=random53();
                                 T.x=360+randx*4300;// this is to generate xc for the simulated image
                                 randy=random53();
                                 T.y=360+randy*4300;

                                for(i=0;i<big_part_data[big_part_no1][1];i++)
                                        for (j=0;j<big_part_data[big_part_no1][2];j++)
                                         {
                                                if (big_part_img[i+j*big_part_data[big_part_no1][1]]==255)
                                                {

                                                        xx=i-big_part_data[big_part_no1][3];
                                                        yy=j-big_part_data[big_part_no1][4];
```

```
                                                        // if ((yy/xx)<0)
                                                        //      theta=1;
                                                         radius=sqrt(xx*xx+yy*yy);
                                                         if (xx==0)
                                                         {
                                                                 theta=Pi/2;
                                                                 if (yy<0) theta=-1*theta;
                                                         }
                                                         else

theta=atan(float(yy)/float(xx));
                                                         if (xx<0) theta = theta + Pi;

                                                         xx=              floor            (
(radius*(cos(theta+alpha)))+.5); // floor -> to round float to the nearest integer
                                                         yy=floor                          (
(radius*(sin(theta+alpha)))+.5);
                                                         if
(imgdata[(T.x+xx)+(T.y+yy)*imageWidth]==255)
                                                         {
                                                                 finish=0;
                                                                 break;
                                                         }
                                                 }
                                         }
                                        if (finish==1) break;
                        }
                                big_part_data_2[big_part_no1].angle=alpha;
                                big_part_data_2[big_part_no1].x=T.x;
                                big_part_data_2[big_part_no1].y=T.y;
                //              fprintf(angle_file,"%f\n",alpha*180/Pi);
                        for(i=0;i<big_part_data[big_part_no1][1];i++)
                                 for (j=0;j<big_part_data[big_part_no1][2];j++)
                                 {
                                        if
(big_part_img[i+j*big_part_data[big_part_no1][1]]==255)
                                        {

                                                xx=i-big_part_data[big_part_no1][3];
                                                yy=j-big_part_data[big_part_no1][4];
                                               // if ((yy/xx)<0)
                                               //      theta=1;
                                                radius=sqrt(xx*xx+yy*yy);
                                                if (xx==0)
                                                {
                                                        theta=Pi/2;
```

```
                                        if (yy<0) theta=-1*theta;
                                }
                                else
                                        theta=atan(float(yy)/float(xx));
                                if (xx<0) theta = theta + Pi;

                                xx= floor ( (radius*(cos(theta+alpha)))+.5);
// floor -> to round float to the nearest integer

                                yy=floor ( (radius*(sin(theta+alpha)))+.5);


imgdata[(T.x+xx)+(T.y+yy)*imageWidth]=255;
                                radius=radius*(cos(theta+alpha));
//                              overall_pixel_count++;
                        }
                }


                        for ( i=T.x-
big_part_data[big_part_no1][1]/2;i<=T.x+big_part_data[big_part_no1][1]/2;i++)
                                for (j=T.y-
big_part_data[big_part_no1][1]/2;j<=T.y+big_part_data[big_part_no1][1]/2;j++)
                                {
                                        if ( (imgdata[i+j*imageWidth]==0) &&
(imgdata[(i-1)+j*imageWidth]==255) && (imgdata[i+(j-1)*imageWidth]==255) &&

(imgdata[(i+1)+j*imageWidth]==255) && (imgdata[i+(j+1)*imageWidth]==255) )
                                                imgdata[i+j*imageWidth]=255;    //
binfill.....


                                if ( imgdata[i+j*imageWidth]==255 )
                                {

px_prt_info[i+j*imageWidth].xc=T.x;

px_prt_info[i+j*imageWidth].yc=T.y;

px_prt_info[i+j*imageWidth].ip=original_nop+big_particles;

prtcl[original_nop+big_particles].area++;
                                        overall_pixel_count++;
                                }

                }
```

```
                }  // end of the for loop for the big particles.. all the big particles are
done.....

        errfile=fopen("error-file.txt","a");
//      fprintf(errfile,"Aa=%f\n",Aa);
        fprintf( errfile,"%d\t",fno);
        for ( i=1;i<=type_of_ellp;i++)
                fprintf(errfile,"%d\t%d\t%0.0f\t",siza[i],sizb[i],freq[i]);
        fprintf(errfile,"%0.3f\t %0.3f\t\t",Aa,lower_limit);
        fclose (errfile);
        pmt[fno].x=siza[1];
        pmt[fno].y=sizb[1];
        pmt[fno].Aa=Aa;
        pmt[fno].lwr_lmt=lower_limit;


  itoa (fno,n1,10);
  strcpy (str,imagename);
        if (no_sims>1) { strcat (str,"-");       strcat (str,n1); }
   alpha_rot=alpha_rot*180/Pi;
        itoa (alpha_rot,n1,10);
        strcat (str,"-rot-");strcat (str,n1);
   alpha_rot=alpha_rot*Pi/180;
                strcpy (lpath_name,str);
        strcat (lpath_name ,"-lineal-path.txt");



        strcpy (str2,str);
        strcat (str2,".txt");
        no_of_ellipse=0;
        if (Aa>0) {
                narea=1;

        {

                fscanf( stream, "%lf %lf %lf",&siza[i], &sizb[i], &freq[i]);
                freq_sum += freq[i];
                narea=narea+siza[i]*sizb[i]*freq[i]/4;
                //cout << freq << endl;
        }
```

```
        Ntot=long (freq_sum*Area*Aa/Pi/narea);

   size_array = new elipse_size[Ntot+100];

//      stream = fopen( fname1, "rt");
        i=0;
        mean_sizea=0;
        mean_sizeb=0;
//      while (!feof(stream))
        for (k=1;k<=type_of_ellp;k++)
        {
//              fscanf( stream, "%lf %lf %lf",&siza, &sizb, &freq);
                for (j=0;j<freq[k]*Ntot/freq_sum;j++)
                {
                        size_array[i].elipa=siza[k];
                        size_array[i++].elipb=sizb[k];
                        mean_sizea +=siza[k];
                        mean_sizeb +=sizb[k];
                }
        }

    lpath=fopen (lpath_name, "w");
        fprintf(lpath,"file is %d \nx=%d y=%d
\nAa=%0.3f\nAlpha_rot=%0.1f",fno,siza[1],sizb[1],Aa,alpha_rot*180/Pi);
        fclose(lpath);

        Ntot=i; // based on the size distribution the Ntot changes slightly
        mean_sizea /= Ntot;
    mean_sizeb /= Ntot;


        //randomize the size distribution
        SimID = (unsigned)time( NULL );
        //SimID = 1003;
        seedMT(SimID);
//      double temp_sizea, temp_sizeb;
//      long ran_n;
        for ( i =0;i<Ntot;i++)
        {
                temp_sizea=size_array[i].elipa;
                temp_sizeb=size_array[i].elipb;
                ran_n =long(Ntot*random53());
                size_array[i]=size_array[ran_n];
                size_array[ran_n].elipa=temp_sizea;
                size_array[ran_n].elipb=temp_sizeb;
```

```
        }

        //"randomize the size distribution"Completed


        Na=Ntot/Area;


//      double cell_size_x, cell_size_y, ratioab;

   ratioab=double(Lx)/double(Ly);
//   Ly=Lx=sqrt(Area);
        //Lx=sqrt(Area*ratioab);
   //Ly=Lx/ratioab;
        cell_size_x = sqrt(in_one_cell/Na*ratioab);
        cell_size_y = sqrt(in_one_cell/Na/ratioab);
   //cell_size_y=cell_size_x = sqrt(in_one_cell/Na);

//      long lx,ly; // lx, ly are the no. of cells in x and y directions
        lx= long (Lx/cell_size_x);
        ly= long (Ly/cell_size_y);



        matrix_2D map_2D(lx,ly); // map_2d is the simulation space

        cell_size_x= Lx/lx;
        cell_size_y= Ly/ly;


        // you can seed with any uint32, but the best are odds in 0..(2^32 - 1)
        SimID = (unsigned)time( NULL );
        //SimID = 1003;
        seedMT(SimID);

        //FILE *ch;ch=fopen("check5.txt","w");
//      double rac,rbc,xc,yc; //current variables
        for ( w=0; w<Ntot; w++)
        {
                bool not_intersect = false;
                long ic,jc;
                rac=size_array[w].elipa/2; //current radius
                rbc=size_array[w].elipb/2;

                while (not_intersect==false)
                {
```

```
                    not_intersect = true;
                    xc=lx*random53(); // this gives a random real number (double) between 0 and lx
                    yc=ly*random53(); // this gives a random real number (double) between 0 and ly

            //      fprintf(ch," %f          %f\n",xc,yc);


                    ic=long (floor(xc));
                    jc=long (floor(yc));

                    for ( i=0;(i<=2)&& not_intersect;i++)
                            for ( j=0;(j<=2)&& not_intersect;j++)
                            not_intersect = !map_2D.element(i+ic-1,j+jc-1).intersect( ( xc-ic+(1-i) )*cell_size_x,( yc-jc+(1-j) )*cell_size_y,rac, rbc);

            }
            map_2D.element(ic,jc).add(          (xc-ic)*cell_size_x,          (yc-jc)*cell_size_y,rac,rbc); // this adds a local

                                                        // coordinate of the cell ic, jc

                                                                // in real scale
    cout << w+1<< endl;


    }
    *noe=Ntot;
    stream=fopen("output.txt","w");

     for (i=0;i<lx;i++)
    {
            for(j=0;j<ly;j++)
            {
                    map_2D.element(i,j).pr_list_2D                          (stream, i*cell_size_x,j*cell_size_y,ellp);
            }
    }
    fclose(stream);
    delete size_array;
    stream = fopen( "output.txt", "rt");
//      out123=fopen("output123.txt","w");  // this one looks dubious ..... 5/15
    i=1;
    while (!feof(stream))
```

```
	{

		fscanf( stream, "%lf %lf %lf %lf",&ex, &ey, &ea, &eb);
		if (feof(stream))
			break;
		ellp[i].x=ex;
		ellp[i].y=ey;
		ellp[i].a=ea;
		ellp[i].b=eb;

		i++;
		//cout << freq << endl;
	}

	fclose( stream );

	stream = fopen( "output.txt", "a");
	fprintf(stream,"%d\t%f\t\t%f\t%d\n",SimID,lx*cell_size_x,ly*cell_size_y, Ntot);
	fprintf(stream,"%f\t%3.15f\t%f\t%f",Aa,Na,mean_sizea, mean_sizeb);
	fclose(stream);



no_of_ellipse=*noe;

	}

	x_s=imageWidth;y_s=imageHeight;
//	  ch=fopen("ch3.txt","w");
//	  ch8=fopen("ch8.txt","w"); // 5/5
//	long SimID;


	outcheck=fopen("outcheck.txt","w");
	fprintf(outcheck,"this is file no. %d---------and no. of ellipse =%d-------
\n",fno,no_of_ellipse);
	//SimID = 1003;
	SimID = (unsigned)time( NULL );
	seedMT(SimID);
//	k=50*random53();
//	imgdata[k+k*imageWidth]=255;
	for (i=1;i<=no_of_ellipse;i++)
```

```
                fprintf(outcheck,"\t%d\t\t%d\t\t%d\t\t%d\n",        ellp[i].x,      ellp[i].y,
ellp[i].a,ellp[i].b);
        fclose(outcheck);

 borderx=imageWidth;
 bordery=imageHeight;   ////////////////////////////////////////////////////////////

 frame_area=borderx*bordery;
         af=0;sum_area_in=0;tot_area_out=0;
//       %vol_frac=.2945;  % it generates vol frac = .273
//       %vol_frac=.225;  % for cv<40

//      %lp=2200;    % no. of additional particles needed to compensate for the
overlapping
//      lp=round(2200*vol_frac/.26);
//      tot_p=round(9*size((bonds22),1)*vol_frac/.26)+lp; % total no. of particles
//      jpeg=1; % =1 if want .jpg output
                imgname="31sim_cv-296_5";
//              % imgname='11sim-1.jpg';
         ellipse_boundary=0 ; //% =1 if want boundary
        particle_outside=1;
        particle_inside=0;
        points=0;
        xtrap=0;current_size=original_nop;
        tot_area=0.0;
/*      %upper_limit=.442;   %81
        %lower_limit=.438;*/
/*float upper_limit=.298;   //%11
float   lower_limit=.294;*/

/*float upper_limit=.47;   //%11
float   lower_limit=.46;*/
        upper_limit=lower_limit+.0005;   //%11

ellipse_boundary=0;

 for(i=1;i<=1000;i++) counter_check[i]=0;
  tot_elp_area=0.0; sum_in_ar_fr=0.0;out_ar_frac=0.0;

//%----------------- particles inside start -------------------------


alpha=0;tot_no_part=0;

//if particle_inside==1
//%elp_center;
```

```
fp=fopen("check3.txt","w"); // 5/5 fp2=fopen("check4.txt","w");
  ip=1 ;
  np=0;
  check=0;check1=0;
//color='b';
  finish=0;
SimID = (unsigned)time( NULL );
                    seedMT(SimID);
for (cp=1;cp<=no_of_ellipse;cp++)
{
          area_elp=ellp[cp].a*ellp[cp].b*Pi;
          tot_elp_area=tot_elp_area+area_elp;

}


for (cp=1;cp<=no_of_ellipse;cp++){
/
          area_elp=ellp[cp].a*ellp[cp].b*Pi;

          f_length=sqrt((ellp[cp].a*ellp[cp].a)-(ellp[cp].b*ellp[cp].b));
          F1.x=ellp[cp].x-f_length;

          F1.y=ellp[cp].y;
          F2.x=ellp[cp].x+f_length;
          F2.y=ellp[cp].y;

          particle_counter=0;
          temp_in_pixel_cnt=0.0;
          in_pixel_count=0.0;
          alpha=(alpha_max*random53())*Pi/180;
          if (cp==66)
                    cp=66;


          int       area_particle=0;

          int flag=1;
          while (flag==1)
          {
                    if (tot_no_part==4596)
                              xx=1;
                    randx=random53();
                    T.x=ellp[cp].x-ellp[cp].a+randx*ellp[cp].a*2;
                    randy=random53();
                    T.y=ellp[cp].y-ellp[cp].b+randy*ellp[cp].b*2;
```

```
xx=T.x; yy=T.y;

if (xx<0)
        xx=xx+borderx;


if (xx>borderx)
        xx=xx-borderx;

if (yy<0)
        yy=yy+bordery;

if (yy>bordery)
        yy=yy-bordery;
d1= sqrt((T.x-F1.x)*(T.x-F1.x)+(T.y-F1.y)*(T.y-F1.y));
d2= sqrt((T.x-F2.x)*(T.x-F2.x)+(T.y-F2.y)*(T.y-F2.y));
if ((d1+d2)>2*ellp[cp].a)   continue;

if (imgdata[xx+yy*imageWidth]==255) continue;

for (i=1;i<=prtcl[ip].area;i++)
{


                                xx=prtcl[ip].fillx[i]-prtcl[ip].xc+T.x;
                                yy=prtcl[ip].filly[i]-prtcl[ip].yc+T.y;
                                if (xx<0)
                                        xx=xx+borderx;


                                if (xx>borderx)
                                        xx=xx-borderx;


                                if (yy<0)
                                        yy=yy+bordery;


                                if (yy>bordery)
                                        yy=yy-bordery;
                                if (imgdata[xx+yy*imageWidth]!=255)
                                {
                                        temp_in_pixel_cnt++;
```

```
                                        }
}
area_frac=temp_in_pixel_cnt/area_elp;




if (particle_counter<= 1000) //original_nop)
{
        if (area_frac>upper_limit)
        {

                //area_particle=area_particle-prtcl[ip].area;
                particle_counter++;
        //      ip++;
        //      if (ip>original_nop) ip=1;
                temp_in_pixel_cnt=in_pixel_count;
                continue;
        }
        if (overlap_check==1)
        {
                        out1=0;
                        no_ovrlpd_prt=0;
                        ovrlp_counter=0.0;
                        for (i=1;i<=prtcl[ip].area;i++)
                        {

                                xx=prtcl[ip].fillx[i]-prtcl[ip].xc+T.x;
                                yy=prtcl[ip].filly[i]-prtcl[ip].yc+T.y;


                                if (xx<0)
                                        xx=xx+borderx;


                                if (xx>borderx)
                                        xx=xx-borderx;


                                if (yy<0)
                                        yy=yy+bordery;


                                if (yy>bordery)
                                        yy=yy-bordery;
```

```
                                        if (imgdata[xx+yy*imageWidth]==255)
                                        {
                                                ovrlp_counter++;
                                                new_prt=1;
                                                if (no_ovrlpd_prt==0)
                                                {
                                                        no_ovrlpd_prt=1; // no. of
overlapped particles

        ovrlpd_prt[1].xc=px_prt_info[xx+yy*imageWidth].xc;

        ovrlpd_prt[1].yc=px_prt_info[xx+yy*imageWidth].yc;

        ovrlpd_prt[1].prt_area=prtcl[px_prt_info[xx+yy*imageWidth].ip].area;

        ovrlpd_prt[1].ovrlpd_pixels=px_prt_info[xx+yy*imageWidth].ovrlpd_pixels;
                                                }

                                                for (k=1;k<=no_ovrlpd_prt;k++)
                                                {
                                                        if
(px_prt_info[xx+yy*imageWidth].xc==ovrlpd_prt[k].xc                             &&
px_prt_info[xx+yy*imageWidth].yc==ovrlpd_prt[k].yc)
                                                        {

        ovrlpd_prt[k].ovrlpd_pixels=ovrlpd_prt[k].ovrlpd_pixels+1;
                                                                new_prt=0;

        px_prt_info[xx+yy*imageWidth].ovrlpd_pixels=px_prt_info[xx+yy*imageWidth
].ovrlpd_pixels+1;


                                                        }
                                                }
                                                if (new_prt==1)
                                                {
                                                        no_ovrlpd_prt++;

        ovrlpd_prt[no_ovrlpd_prt].xc=px_prt_info[xx+yy*imageWidth].xc;

        ovrlpd_prt[no_ovrlpd_prt].yc=px_prt_info[xx+yy*imageWidth].yc;

        ovrlpd_prt[no_ovrlpd_prt].prt_area=prtcl[px_prt_info[xx+yy*imageWidth].ip].ar
ea;
```

```
        ovrlpd_prt[no_ovrlpd_prt].ovrlpd_pixels=1+px_prt_info[xx+yy*imageWidth].ovr
lpd_pixels;

                                                }
                                                if   ((ovrlp_counter/prtcl[ip].area)   >
overlap_ratio )
                                                {

                                                        out1=1;
                                                        break;
                                                }
                                                for (k=1;k<=no_ovrlpd_prt;k++)
                                                {
                                                        if
((ovrlpd_prt[k].ovrlpd_pixels/ovrlpd_prt[k].prt_area) > overlap_ratio )
                                                        {
                                                                out1=1;
                                                                //particle_counter++;
                                                        //      ip++;
                                                        //      if    (ip>original_nop)
ip=1;
                                                                break;
                                                        }
                                                }

                                                if (out1==1) break;
                                                break;
                                        }         //          end          for          "if
(imgdata[xx+yy*imageWidth]==255) "
                                        } // end for particle pixels



                                        if (out1==1)
                                        {
//                                              fprintf(ch6,"overlapd ar fr = %f    %f%d\n",
(ovrlpd_prt[k].ovrlpd_pixels/ovrlpd_prt[k].prt_area),(ovrlp_counter/prtcl[ip-
1].area),out1);
                                                continue;
                                        }


                                /*      if (out==1) {
                                                fprintf(ch6,"overlapd ar fr = %f    %f%d\n",
(ovrlpd_prt[k].ovrlpd_pixels/ovrlpd_prt[k].prt_area),(ovrlp_counter/prtcl[ip].area),out);
```

```
                        continue;
                        }*/
                }
                for (i=1;i<=prtcl[ip].area;i++)
                {


                                        xx=prtcl[ip].fillx[i]-prtcl[ip].xc+T.x;
                                        yy=prtcl[ip].filly[i]-prtcl[ip].yc+T.y;

                                        if (xx<0)
                                                xx=xx+borderx;


                                        if (xx>borderx)
                                                xx=xx-borderx;


                                        if (yy<0)
                                                yy=yy+bordery;


                                        if (yy>bordery)
                                                yy=yy-bordery;
                                        if (imgdata[xx+yy*imageWidth]!=255)
                                        {
                                                if (only_points==1)

                                                {
                                                        dist_x=abs(prtcl[ip].fillx[i]-
prtcl[ip].xc);
                                                        dist_y=abs(prtcl[ip].filly[i]-
prtcl[ip].yc);
                                                //      if
((prtcl[ip].fillx[i]==prtcl[ip].xc)&&(prtcl[ip].filly[i]==prtcl[ip].yc))
imgdata[xx+yy*imageWidth]=255; // changed 7/19 -- only to get the points
                                                   if       ((dist_x<4)&&(dist_y<4))
imgdata[xx+yy*imageWidth]=255; // changed 7/19 -- only to get the points
                                                }                                       else
imgdata[xx+yy*imageWidth]=255;

                                                in_pixel_count++;
                                                overall_pixel_count++  ;
                                        }
                                        else
```

```
                                        {

        px_prt_info[xx+yy*imageWidth].ovrlpd_pixels=px_prt_info[xx+yy*imageWidth
].ovrlpd_pixels+1;
                                        //      imgdata[xx+yy*imageWidth]=0;
                                        }


                                    //      imgdata[xx+yy*imageWidth]=255;


                                        px_prt_info[xx+yy*imageWidth].xc=T.x;
                                        px_prt_info[xx+yy*imageWidth].yc=T.y;
                                        px_prt_info[xx+yy*imageWidth].ip=ip;
                    }
                    counter_check[ip]=counter_check[ip]+1;



                tot_no_part++;
                part_info[tot_no_part].ip=ip;
                part_info[tot_no_part].xc=T.x;
                part_info[tot_no_part].yc=T.y;
                part_info[tot_no_part].angle=alpha+prtcl[ip].angle;


        fprintf(part_info_file,"%d\t\t%d\t\t%d\t\t%d\t\t%d\t%d\n",tot_no_part,T.x,T.y,ip,
prtcl[ip].xc,prtcl[ip].yc);


                // particle size distribution matching
                for(i=1;i<=no_of_bin;i++)
                {
                        if (prtcl[ip].area<=prt_size_bin[i][0])
                        {
                                prt_size_bin[i][3]++;
                                break;
                        }
                }
                ip++;
                if (ip>original_nop) ip=1;
                for(i=1;i<=no_of_bin;i++)
                {
                        if (prtcl[ip].area<=prt_size_bin[i][0])
                                break;
                }
```

```
		out=0;
		while (out<=original_nop)
		{
			out++;

			if (prt_size_bin[i][3]+1>prt_size_bin[i][2])
			{
				ip++;
				if (ip>original_nop) ip=1;
				for(i=1;i<=no_of_bin;i++)
				{
					if (prtcl[ip].area<=prt_size_bin[i][0])
						break;
				}
			}
			else
				break;
		}
		// particle size distribution matching

			temp_in_pixel_cnt=in_pixel_count;
			temp_overall_pixel_cnt=overall_pixel_count;
			area_frac=in_pixel_count/area_elp;
			overall_ar_frac=overall_pixel_count/frame_area;
			if (overall_ar_frac>=vol_frac)
			{
				finish=1;
				break;
			}
			if (area_frac>=lower_limit) flag=0;


}
		else
		{
				break;
		}


   }//end;// end for "while" loop
   area_frac=in_pixel_count/area_elp;
   sum_in_ar_fr=sum_in_ar_fr+area_frac;

   if (finish==1)
		break;
   //end;
```

```
}


fprintf(fp,"%d %f\n",check,overall_ar_frac );
fclose(fp);


inside_ar_frac=sum_in_ar_fr/no_of_ellipse;

 ch9=fopen("ch9.txt","w");
fprintf(ch9,"\noverall area frac = %f",overall_ar_frac);


overall_ar_frac=overall_pixel_count/frame_area;
 ch4=fopen("ch4.txt","w");

fprintf(ch4,"overall area frac = %f   inside ar fr =  %f\n",overall_ar_frac,inside_ar_frac);
int flag,area_temp;

if (particle_outside==1) {

        out1=0;
while (overall_ar_frac<vol_frac) {

        if (overall_ar_frac>.07)
                out1=0;
   flag=1;
        randx=random53();
        randy=random53();


        if (out1==0)
        {
                alpha=(alpha_max*random53())*Pi/180;
                alpha=alpha-(alpha_max*Pi/180)/2;.
        }
        if (out1==1)
        {
                out1=1;
        }

        orig_prt_area=prtcl[ip].area;
        xp=borderx*randx; yp=bordery*randy;
        xx=xp;yy=yp;
        T.x=xp; T.y=yp;//T.x=4950;T.y=2804;
```

```
   //for cp=1:no_of_ellipse;
        for (cp=1;cp<=no_of_ellipse;cp++)
        {


                        f_length=sqrt((ellp[cp].a*ellp[cp].a)-(ellp[cp].b*ellp[cp].b));
                        F1.x=ellp[cp].x-f_length;

                        F1.y=ellp[cp].y;
                        F2.x=ellp[cp].x+f_length;
                        F2.y=ellp[cp].y;
                        T.x=xp; T.y=yp;
                        d1= sqrt((T.x-F1.x)*(T.x-F1.x)+(T.y-F1.y)*(T.y-F1.y));
                        d2= sqrt((T.x-F2.x)*(T.x-F2.x)+(T.y-F2.y)*(T.y-F2.y));
                        if ((d1+d2)<2*ellp[cp].a)   flag=0;

//      if (d1+d2)<2*elp_center(cp,3)


        }
   if (flag==0)
        continue;



        if (alpha != 0 ) { // this portion is to be used only when random rotation is
required ..
        if (overlap_check==1)
                        {
                                out1=0;
                                no_ovrlpd_prt=0;
                                ovrlp_counter=0.0;
                                for (i=1;i<=prtcl[ip].area;i++)
                                {
                                        xx=prtcl[ip].fillx[i]-prtcl[ip].xc;//+T.x;
                                        yy=prtcl[ip].filly[i]-prtcl[ip].yc;//+T.y;

                                        radius=sqrt(xx*xx+yy*yy);
                                                if (xx==0)
                                                {
                                                        theta=Pi/2;
                                                        if (yy<0) theta=-1*theta;
                                                }
                                                else

theta=atan(float(yy)/float(xx));
```

```
                                        if (xx<0) theta = theta + Pi;

                                        xx=          T.x+floor          (
(radius*(cos(theta+alpha)))+.5); // floor -> to round float to the nearest integer
                                        yy=T.y+floor                    (
(radius*(sin(theta+alpha)))+.5);


                                    if (xx<0)
                                        xx=xx+borderx;


                                    if (xx>=borderx)
                                        xx=xx-borderx;


                                    if (yy<0)
                                        yy=yy+bordery;


                                    if (yy>=bordery)
                                        yy=yy-bordery;
                                    if (imgdata[xx+yy*imageWidth]==255)
                                    {
                                        out1=1;
                                        break;
                                    }

                                    temp_layer[xx+yy*imageWidth]=255;
                                }

                                if (out1==1)
                                {
                                    for(xx=T.x-
prtcl_window/2;xx<=T.x+prtcl_window/2;xx++)
                                    for                                 (yy=T.y-
prtcl_window/2;yy<=T.y+prtcl_window/2;yy++)
                                    {

                                        i=xx;j=yy;

                                        if (i<0)
                                        i=i+borderx;


                                        if (i>=borderx)
```

```
                                        i=i-borderx;


                                    if (j<0)
                                        j=j+bordery;


                                    if (j>=bordery)
                                        j=j-bordery;

                                    temp_layer[i+j*imageWidth]=0;
                                }

                                continue;
                            }

                            area_temp=prtcl[ip].area;
                            prtcl[ip].area=0; // area is changed after the particle
is rotated.....


// put the rotated particle on the "temp layer" .. and the next step is to binfil it ....

//
//
//
                            for(xx=T.x-
prtcl_window/2;xx<=T.x+prtcl_window/2;xx++)
                                for                                 (yy=T.y-
prtcl_window/2;yy<=T.y+prtcl_window/2;yy++)
                                {
                                    i=xx;j=yy;
                                    if (i<0)
                                    i=i+borderx;

                                    if (i>=borderx)
                                        i=i-borderx;


                                    if (j<0)
                                        j=j+bordery;


                                    if (j>=bordery)
                                        j=j-bordery;
```

```
                                        lft=i-1;rht=i+1;
                                        up=j-1;dwn=j+1;
                                        if (lft<0)
                                        lft=lft+borderx;



                                        if (rht>=borderx)
                                                rht=rht-borderx;

                                        if (up<0)
                                                up=up+bordery;
                                        if (dwn>=bordery)
                                                dwn=dwn-bordery;


                                        if                                              (
(temp_layer[i+j*imageWidth]==0)  &&  (temp_layer[lft+j*imageWidth]==255)  &&
(temp_layer[i+up*imageWidth]==255) &&

(temp_layer[rht+j*imageWidth]==255) && (temp_layer[i+dwn*imageWidth]==255) )
                                                 temp_layer[i+j*imageWidth]=255;
// binfill.....

                                        if
(temp_layer[i+j*imageWidth]==255)
                                        {
                                                prtcl[ip].area++;
                                        }

                                }

                                //binfill is done ....

                        for(i=T.x-
prtcl_window/2;i<=T.x+prtcl_window/2;i++)
                                for                                             (j=T.y-
prtcl_window/2;j<=T.y+prtcl_window/2;j++)
                                {

                                        xx=i;yy=j;
                                        if (xx<0)
                                        xx=xx+borderx;


                                        if (xx>=borderx)
```

```
                    xx=xx-borderx;

                if (yy<0)
                    yy=yy+bordery;

                if (yy>=bordery)
                    yy=yy-bordery;

                if
(temp_layer[xx+yy*imageWidth]==255)
                {

                    if
(imgdata[xx+yy*imageWidth]==255)
                    {
                        ovrlp_counter++;
                        new_prt=1;
                        if (no_ovrlpd_prt==0)
                        {
        no_ovrlpd_prt=1; // no. of overlapped particles
        ovrlpd_prt[1].xc=px_prt_info[xx+yy*imageWidth].xc;
        ovrlpd_prt[1].yc=px_prt_info[xx+yy*imageWidth].yc;
        ovrlpd_prt[1].prt_area=prtcl[px_prt_info[xx+yy*imageWidth].ip].area;
        ovrlpd_prt[1].ovrlpd_pixels=px_prt_info[xx+yy*imageWidth].ovrlpd_pixels;
                        }

                        for
(k=1;k<=no_ovrlpd_prt;k++)
                        {
                            if
(px_prt_info[xx+yy*imageWidth].xc==ovrlpd_prt[k].xc                    &&
px_prt_info[xx+yy*imageWidth].yc==ovrlpd_prt[k].yc)
                            {
        ovrlpd_prt[k].ovrlpd_pixels=ovrlpd_prt[k].ovrlpd_pixels+1;
        new_prt=0;
```

```
        px_prt_info[xx+yy*imageWidth].ovrlpd_pixels=px_prt_info[xx+yy*imageWidth
].ovrlpd_pixels+1;


                                                                    }
                                                                }
                                                                if (new_prt==1)
                                                                {

        no_ovrlpd_prt++;

        ovrlpd_prt[no_ovrlpd_prt].xc=px_prt_info[xx+yy*imageWidth].xc;

        ovrlpd_prt[no_ovrlpd_prt].yc=px_prt_info[xx+yy*imageWidth].yc;

        ovrlpd_prt[no_ovrlpd_prt].prt_area=prtcl[px_prt_info[xx+yy*imageWidth].ip].ar
ea;

        ovrlpd_prt[no_ovrlpd_prt].ovrlpd_pixels=1+px_prt_info[xx+yy*imageWidth].ovr
lpd_pixels;

                                                                }
                                                                if
((ovrlp_counter/prtcl[ip].area) > overlap_ratio)
                                                                {

        //particle_counter++;
                                                                //      ip++;
                                                                //      if
(ip>original_nop) ip=1;
                                                                        out1=1;
                                                                        break;
                                                                }
                                                                for
(k=1;k<=no_ovrlpd_prt;k++)
                                                                {
                                                                        if
((ovrlpd_prt[k].ovrlpd_pixels/ovrlpd_prt[k].prt_area) > overlap_ratio )
                                                                        {

        out1=1;

        //particle_counter++;
                                                                        //      ip++;
```

```
                                                        //      if
(ip>original_nop) ip=1;
                                                                break;
                                                        }
                                                }

                                                if (out1==1) break;


                                                } // end for "if
(imgdata[xx+yy*imageWidth]==255) "
                                        } // end for particle pixels
                                                }// end of " if
temp_layer[i+j*imageWidth]==255

                                        }


                                if (out1==1)
                                {
                                        for(xx=T.x-
prtcl_window/2;xx<=T.x+prtcl_window/2;xx++)
                                        for                     (yy=T.y-
prtcl_window/2;yy<=T.y+prtcl_window/2;yy++)
                                        {
                                                i=xx;j=yy;



                                                if (i<0)
                                                i=i+borderx;


                                                if (i>=borderx)
                                                        i=i-borderx;


                                                if (j<0)
                                                        j=j+bordery;


                                                if (j>=bordery)
                                                        j=j-bordery;

                                                temp_layer[i+j*imageWidth]=0;
                                        }
```

```
//                                      fprintf(ch6,"overlapd ar fr = %f    %f%d\n",
(ovrlpd_prt[k].ovrlpd_pixels/ovrlpd_prt[k].prt_area),(ovrlp_counter/prtcl[ip-
1].area),out1);
                                        continue;
                                    }


                            /*      if (out==1) {
                                        fprintf(ch6,"overlapd ar fr = %f    %f%d\n",
(ovrlpd_prt[k].ovrlpd_pixels/ovrlpd_prt[k].prt_area),(ovrlp_counter/prtcl[ip].area),out);
                                    continue;
                                    }*/
                        }
//                      for (i=1;i<=prtcl[ip].area;i++)


                //                fprintf(angle_file,"%f\n",alpha*180/Pi);
                tot_no_part++;
                part_info[tot_no_part].ip=ip;
                part_info[tot_no_part].xc=T.x;
                part_info[tot_no_part].yc=T.y;
                part_info[tot_no_part].angle=alpha+prtcl[ip].angle;



        fprintf(part_info_file,"%d\t%d\t%d\t%d\t%d\t%d\n",tot_no_part,T.x,T.y,ip,prtcl[i
p].xc,prtcl[ip].yc);


        // particle transfer from 'temp layer' to imgdata ....
                        for(i=T.x-prtcl_window/2;i<=T.x+prtcl_window/2;i++)
                        for (j=T.y-prtcl_window/2;j<=T.y+prtcl_window/2;j++)
                        {

                                        xx=i;yy=j;

                                        if (xx<0)
                                                xx=xx+borderx;


                                        if (xx>=borderx)
                                                xx=xx-borderx;


                                        if (yy<0)
```

```
                                        yy=yy+bordery;


                                    if (yy>=bordery)
                                        yy=yy-bordery;
                                    if (temp_layer[xx+yy*imageWidth]==255)
                                    {
                                        if
(imgdata[xx+yy*imageWidth]!=255)
                                        {
                                            if (only_points==1)

                                            {

        dist_x=abs(prtcl[ip].fillx[i]-prtcl[ip].xc);

        dist_y=abs(prtcl[ip].filly[i]-prtcl[ip].yc);
                                            //      if
((prtcl[ip].fillx[i]==prtcl[ip].xc)&&(prtcl[ip].filly[i]==prtcl[ip].yc))
imgdata[xx+yy*imageWidth]=255; // changed 7/19 -- only to get the points
                                                if
((dist_x<4)&&(dist_y<4)) imgdata[xx+yy*imageWidth]=255;
                                            }                           else
imgdata[xx+yy*imageWidth]=255;
                                            in_pixel_count++;

                                            overall_pixel_count++  ;
                                        }
                                        else
                                        {

        px_prt_info[xx+yy*imageWidth].ovrlpd_pixels=px_prt_info[xx+yy*imageWidth
].ovrlpd_pixels+1;
                                        //
        imgdata[xx+yy*imageWidth]=0;
                                        }


                                    //      imgdata[xx+yy*imageWidth]=255;



        px_prt_info[xx+yy*imageWidth].xc=T.x;

        px_prt_info[xx+yy*imageWidth].yc=T.y;
```

```
        px_prt_info[xx+yy*imageWidth].ip=ip;
                                        }
                        }


                        for(xx=T.x-prtcl_window/2;xx<=T.x+prtcl_window/2;xx++)
                        for (yy=T.y-prtcl_window/2;yy<=T.y+prtcl_window/2;yy++)
                        {
                                        i=xx;j=yy;



                                        if (i<0)
                                        i=i+borderx;


                                        if (i>=borderx)
                                                i=i-borderx;


                                        if (j<0)
                                                j=j+bordery;


                                        if (j>=bordery)
                                                j=j-bordery;

                                        temp_layer[i+j*imageWidth]=0;
                        }

                        prtcl[ip].area=area_temp;

                        counter_check[ip]=counter_check[ip]+1;
                        prtcl[ip].area=orig_prt_area;
}


if (alpha == 0.0) {

        if (overlap_check==1)
                        {
                                        out1=0;
                                        no_ovrlpd_prt=0;
                                        ovrlp_counter=0.0;
                                        for (i=1;i<=prtcl[ip].area;i++)
```

```
{
        xx=prtcl[ip].fillx[i]-prtcl[ip].xc+T.x;
        yy=prtcl[ip].filly[i]-prtcl[ip].yc+T.y;
        if (xx<0)
                xx=xx+borderx;


        if (xx>borderx)
                xx=xx-borderx;


        if (yy<0)
                yy=yy+bordery;


        if (yy>bordery)
                yy=yy-bordery;

        if (imgdata[xx+yy*imageWidth]==255)
        {
                ovrlp_counter++;
                new_prt=1;
                if (no_ovrlpd_prt==0)
                {
                        no_ovrlpd_prt=1; // no. of
overlapped particles

        ovrlpd_prt[1].xc=px_prt_info[xx+yy*imageWidth].xc;

        ovrlpd_prt[1].yc=px_prt_info[xx+yy*imageWidth].yc;

        ovrlpd_prt[1].prt_area=prtcl[px_prt_info[xx+yy*imageWidth].ip].area;

        ovrlpd_prt[1].ovrlpd_pixels=px_prt_info[xx+yy*imageWidth].ovrlpd_pixels;
                }

                for (k=1;k<=no_ovrlpd_prt;k++)
                {
                        if
(px_prt_info[xx+yy*imageWidth].xc==ovrlpd_prt[k].xc                     &&
px_prt_info[xx+yy*imageWidth].yc==ovrlpd_prt[k].yc)
                        {

        ovrlpd_prt[k].ovrlpd_pixels=ovrlpd_prt[k].ovrlpd_pixels+1;
                                new_prt=0;
```

```
        px_prt_info[xx+yy*imageWidth].ovrlpd_pixels=px_prt_info[xx+yy*imageWidth
].ovrlpd_pixels+1;
                                                        //
        ovrlpd_prt[k].ovrlpd_pixels=px_prt_info[xx+yy*imageWidth].ovrlpd_pixels;

                                                        }
                                                }
                                                if (new_prt==1)
                                                {
                                                        no_ovrlpd_prt++;

        ovrlpd_prt[no_ovrlpd_prt].xc=px_prt_info[xx+yy*imageWidth].xc;

        ovrlpd_prt[no_ovrlpd_prt].yc=px_prt_info[xx+yy*imageWidth].yc;

        ovrlpd_prt[no_ovrlpd_prt].prt_area=prtcl[px_prt_info[xx+yy*imageWidth].ip].ar
ea;

        ovrlpd_prt[no_ovrlpd_prt].ovrlpd_pixels=1+px_prt_info[xx+yy*imageWidth].ovr
lpd_pixels;
                                                //
        px_prt_info[xx+yy*imageWidth].ovrlpd_pixels=px_prt_info[xx+yy*imageWidth
].ovrlpd_pixels+1;
                                                }
                                                if   ((ovrlp_counter/prtcl[ip].area)   >
overlap_ratio)
                                                {
                                                        //particle_counter++;
                                                //      ip++;
                                                //      if (ip>original_nop) ip=1;
                                                        out1=1;
                                                        break;
                                                }
                                                for (k=1;k<=no_ovrlpd_prt;k++)
                                                {
                                                        if
((ovrlpd_prt[k].ovrlpd_pixels/ovrlpd_prt[k].prt_area) > overlap_ratio )
                                                        {
                                                                out1=1;
                                                                //particle_counter++;
                                                        //      ip++;
                                                        //      if    (ip>original_nop)
ip=1;
                                                                break;
                                                        }
```

```
                                                }
//                                              if (out1==1) { //fprintf(ch6,"overlapd
ar          fr          =          %f                                          %f\n",
(ovrlpd_prt[k].ovrlpd_pixels/ovrlpd_prt[k].prt_area),(ovrlp_counter/prtcl[ip].area));
//                                  break;}
                                                if (out1==1) break;

                                            }       //      end     for     "if
(imgdata[xx+yy*imageWidth]==255) "
                                    } // end for particle pixels



                                    if (out1==1)
                                    {
                                            particle_counter++;
                                            continue;
                                    }


                            }
                    tot_no_part++;
                    part_info[tot_no_part].ip=ip;
                    part_info[tot_no_part].xc=T.x;
                    part_info[tot_no_part].yc=T.y;
                    part_info[tot_no_part].angle=prtcl[ip].angle;


        fprintf(part_info_file,"%d\t%d\t%d\t%d\t%d\t%d\n",tot_no_part,T.x,T.y,ip,prtcl[i
p].xc,prtcl[ip].yc);


                    for (i=1;i<=prtcl[ip].area;i++)
                    {


                                            xx=prtcl[ip].fillx[i]-prtcl[ip].xc+T.x;
                                            yy=prtcl[ip].filly[i]-prtcl[ip].yc+T.y;
                                            if (xx<0)
                                                    xx=xx+borderx;


                                            if (xx>borderx)
                                                    xx=xx-borderx;
```

```
                                        if (yy<0)
                                                yy=yy+bordery;


                                        if (yy>bordery)
                                                yy=yy-bordery;
                                        if (imgdata[xx+yy*imageWidth]!=255)
                                        {
                                                imgdata[xx+yy*imageWidth]=255;
// THIS IS WHERE WRE ARE PUTTING PARTICLES OUTSIDE THE ELLIPSE
                                                in_pixel_count++;
                                                overall_pixel_count++  ;
                                        }
                                        else
                                        {

        px_prt_info[xx+yy*imageWidth].ovrlpd_pixels=px_prt_info[xx+yy*imageWidth
].ovrlpd_pixels+1;

                                        }

                                        px_prt_info[xx+yy*imageWidth].xc=T.x;
                                        px_prt_info[xx+yy*imageWidth].yc=T.y;
                                        px_prt_info[xx+yy*imageWidth].ip=ip;




                        }


}
                for(i=1;i<=no_of_bin;i++)
                {
                        if (prtcl[ip].area<=prt_size_bin[i][0])
                        {
                                prt_size_bin[i][3]++;
                                break;
                        }
                }
                ip++;
                if (ip>original_nop) ip=1;
                for(i=1;i<=no_of_bin;i++)
                {
                        if (prtcl[ip].area<=prt_size_bin[i][0])
                                break;
```

```
}
out=0;
while (out<=1000)
{
        out++;
        if (out==999)
                i=3;
        if (prt_size_bin[i][3]+1>prt_size_bin[i][2])
        {
                ip++;
                if (ip>original_nop) ip=1;
                for(i=1;i<=no_of_bin;i++)
                {
                        if (prtcl[ip].area<=prt_size_bin[i][0])
                                break;
                }
        }
        else
                break;
}



        temp_in_pixel_cnt=in_pixel_count;
        temp_overall_pixel_cnt=overall_pixel_count;
        area_frac=in_pixel_count/area_elp;
        overall_ar_frac=overall_pixel_count/frame_area;









  }
}
fclose(ch4);
fprintf(ch9,"\noverall area frac = %f",overall_ar_frac);
fclose(ch9);
fclose(part_info_file);

lpath=fopen (lpath_name, "a");
fprintf (lpath,"\nCluster Intensity=%f      Vv inside clusters=%f \n no. of ellipse=%d
ovrlp=%f angle=%d\n",clust_inten,lower_limit,no_of_ellipse,overlap_ratio,ang);
```

```
        itoa (fno,n1,10);
   strcpy (str,imagename);
        if (no_sims>1) { strcat (str,"-"); strcat (str,n1); }
        strcat (str,".bmp");
        if (alpha_rot==0)
                writeGrayScaleDataToBmpFile(str, imageWidth, imageHeight, imgdata);
for (i=0;i<=20;i++)
{angle_bin_arr[i][1]=0;angle_bin_arr[i][2]=0;angle_bin_arr[i][3]=0;}
for (particle_counter=1;particle_counter<=tot_no_part;particle_counter++)
                {
                        alpha=part_info[particle_counter].angle;
                        i=0;
                        for (k=-90;k<=90;k=k+10)
                        {
                                i++;
                                if ((alpha*180/Pi)<=k) break; //
                        }
                        angle_bin_arr[i][1]++;
                //      alpha_rot=temp_angle;   // Putting the bin-freq for the angles in the
first random image.
                }
i=0;
for (k=-90;k<=90;k=k+10)
{
        i++;
        t=k;
        if ( t<0) {t=-1*t;t=t+10;}

    t=200-t*190/90;
        t=1000; /


        angle_bin_arr[i][2]=floor ((angle_bin_arr[i][1]*t/100)+.5);
}
FILE *ang_bin_arr;
ang_bin_arr=fopen("ang_bin_arr.txt","w");

i=3;
////////////////////////////////// ROTATING THE PARTICLES
//////////////////////////////////////////
if ( alpha_rot!=0) { // skip this step if rotation of the particles is not required ...

//first step is to erase the particle from the actual image 'imgdata'
//          int direction,rot_counter; // to get the sign of the angle, to count the iterations of
rotations applied on the img_data .. now defined up
```

```
//        float theta_rot; // the new angle at which the particle must be placed.. (before checking for the overlap) ..now defined up
        for                                (rot_counter=1;rot_counter<=1;rot_counter++){
angle_file=fopen("angle_file.txt","w");// overall loop for rotating all prtcls
        for                                                                                (
big_particles_counter=1;big_particles_counter<=big_particle_number;big_particles_cou
nter++)
        {
                big_particles=big_particles_counter;


               big_part_no1=big_particles;
               alpha=big_part_data_2[big_part_no1].angle;
        //      alpha = 0;

               itoa (big_part_no1,n1,10);
               strcpy  (big_part_name,"input_big_part-");
               strcat  (big_part_name,n1);
               strcat  (big_part_name,".bmp");


               big_part_img=LoadBitmapFile(big_part_name,&bitmapInfoHeader);

writeGrayScaleDataToBmpFile("temp.bmp",big_part_data[big_part_no1][1],big_part_da
ta[big_part_no1][2],big_part_img);
               big_part_img=LoadBitmapFile("temp.bmp",&bitmapInfoHeader);
               i=3;



                 T.x=big_part_data_2[big_part_no1].x;
                 T.y=big_part_data_2[big_part_no1].y;

                      for(i=0;i<big_part_data[big_part_no1][1];i++)
                              for (j=0;j<big_part_data[big_part_no1][2];j++)
                              {
                                      if
(big_part_img[i+j*big_part_data[big_part_no1][1]]==255)
                                      {

                                              xx=i-big_part_data[big_part_no1][3];
                                              yy=j-big_part_data[big_part_no1][4];
                                             // if ((yy/xx)<0)
                                             //         theta=1;
                                              radius=sqrt(xx*xx+yy*yy);
                                              if (xx==0)
```

```
                                        {
                                                theta=Pi/2;
                                                if (yy<0) theta=-1*theta;
                                        }
                                        else
                                                theta=atan(float(yy)/float(xx));
                                        if (xx<0) theta = theta + Pi;

                                        xx= floor ( (radius*(cos(theta+alpha)))+.5);
// floor -> to round float to the nearest integer

                                        yy=floor ( (radius*(sin(theta+alpha)))+.5);


temp_layer[(T.x+xx)+(T.y+yy)*imageWidth]=255;
                                        radius=radius*(cos(theta+alpha));
//                                      overall_pixel_count++;
                                }
                        }


                        for ( i=T.x-big_part_data[big_part_no1][1]/2;i<=T.x+big_part_data[big_part_no1][1]/2;i++)
                                for (j=T.y-big_part_data[big_part_no1][1]/2;j<=T.y+big_part_data[big_part_no1][1]/2;j++)
                                {
                                        if ( (temp_layer[i+j*imageWidth]==0) && (temp_layer[(i-1)+j*imageWidth]==255) && (temp_layer[i+(j-1)*imageWidth]==255) &&

(temp_layer[(i+1)+j*imageWidth]==255) && (temp_layer[i+(j+1)*imageWidth]==255) )
                                                temp_layer[i+j*imageWidth]=255;
// binfill.....


                                }

                                for ( i=T.x-big_part_data[big_part_no1][1]/2;i<=T.x+big_part_data[big_part_no1][1]/2;i++)
                                        for (j=T.y-big_part_data[big_part_no1][1]/2;j<=T.y+big_part_data[big_part_no1][1]/2;j++)
                                                if (temp_layer[i+j*imageWidth]==255) imgdata[i+j*imageWidth]=0;  // particle erased from img_data
```

```
//                                      //
                //
                                        for                     (                       i=T.x-
big_part_data[big_part_no1][1]/2;i<=T.x+big_part_data[big_part_no1][1]/2;i++)
                                                for                                     (j=T.y-
big_part_data[big_part_no1][1]/2;j<=T.y+big_part_data[big_part_no1][1]/2;j++)
                                                        temp_layer[i+j*imageWidth]=0;
//temp layer is cleaned..

                                        if  (alpha!=0.0)
                                                direction=alpha/fabs(alpha); // its the sign of
the angle (positive or negative)
                                        else
                                                direction=1;
                                         if (fabs(alpha)<alpha_rot)
                                                        theta_rot =0;
                                         else
                                                        theta_rot=alpha-
(direction)*alpha_rot; // before this add the step where the angle of rotation is adjusted

        // for the size, the angle of orientation and aspect ratio.

                                        alpha=
rot_angle(imgdata,big_part_img,theta_rot,big_part_data[big_part_no1][1],big_part_data[
big_part_no1][2],big_part_data[big_part_no1][3],big_part_data[big_part_no1][4],T.x,T.y
,imageWidth,direction,borderx,bordery,alpha);
                                big_part_data_2[big_part_no1].angle=alpha;
                                i=3;



                                for(i=0;i<big_part_data[big_part_no1][1];i++)
                                 for (j=0;j<big_part_data[big_part_no1][2];j++)
                                 {
                                        if
(big_part_img[i+j*big_part_data[big_part_no1][1]]==255)
                                        {

                                                xx=i-big_part_data[big_part_no1][3];
                                                yy=j-big_part_data[big_part_no1][4];
                                               // if ((yy/xx)<0)
                                               //         theta=1;
                                                radius=sqrt(xx*xx+yy*yy);
                                                if (xx==0)
                                                {
                                                        theta=Pi/2;
```

```
                                        if (yy<0) theta=-1*theta;
                                    }
                                    else
                                        theta=atan(float(yy)/float(xx));
                                    if (xx<0) theta = theta + Pi;

                                    xx= floor ( (radius*(cos(theta+alpha)))+.5);
// floor -> to round float to the nearest integer

                                    yy=floor ( (radius*(sin(theta+alpha)))+.5);

                                    if
(imgdata[(T.x+xx)+(T.y+yy)*imageWidth]==255)
                                        k=3;


imgdata[(T.x+xx)+(T.y+yy)*imageWidth]=255;
                                    radius=radius*(cos(theta+alpha));
//                                  overall_pixel_count++;
                                }
                            }

                            for ( i=T.x-
big_part_data[big_part_no1][1]/2;i<=T.x+big_part_data[big_part_no1][1]/2;i++)
                                for (j=T.y-
big_part_data[big_part_no1][1]/2;j<=T.y+big_part_data[big_part_no1][1]/2;j++)
                                {
                                    if ( (imgdata[i+j*imageWidth]==0) &&
(imgdata[(i-1)+j*imageWidth]==255) && (imgdata[i+(j-1)*imageWidth]==255) &&

(imgdata[(i+1)+j*imageWidth]==255) && (imgdata[i+(j+1)*imageWidth]==255) )
                                        imgdata[i+j*imageWidth]=255;    //
binfill.....



                                }
            i=3;


            } //
            i=3;
```

```
//                    ////////////////////////////////////////////////BIG PARTICLES ROTATED ////////////////////////////////////////

//              avg_angle=0.0;
                for (particle_counter=1;particle_counter<=tot_no_part;particle_counter++)
                        avg_angle=avg_angle+fabs(part_info[particle_counter].angle);    // why fabs ? to get the absolute value
                avg_angle=avg_angle/tot_no_part;  // need to calculate the avg angle after each simulation

                for (particle_counter=1;particle_counter<=tot_no_part;particle_counter++)
                {
                        ip=part_info[particle_counter].ip;
                        T.x=part_info[particle_counter].xc;
                        T.y=part_info[particle_counter].yc;
                        alpha=part_info[particle_counter].angle; // start with alpha equal to the actual angle
                        if (alpha==0.0) {fprintf (angle_file,"0\t0\t0\n");continue; } //
                        if (ip==379)
                                ip=379;


                                alpha=part_info[particle_counter].angle-prtcl[ip].angle;    // (final)-(intial) or (output)-(input) i.e the relative movement of the particle

                                // there are two different angles here 1) wrt to the intial input image (which must be used for putting or deleting the particle
                                //                                  2) the actual angle of the particle (which must be used in all the measurements

                        for (i=1;i<=prtcl[ip].area;i++)
                        {
                                xx=prtcl[ip].fillx[i]-prtcl[ip].xc;//+T.x;
                                yy=prtcl[ip].filly[i]-prtcl[ip].yc;//+T.y;



                                radius=sqrt(xx*xx+yy*yy);
                                        if (xx==0)
                                        {
                                                theta=Pi/2;
                                                if (yy<0) theta=-1*theta;
                                        }
                                        else
```

```
                                        theta=atan(float(yy)/float(xx));
                                    if (xx<0) theta = theta + Pi;

                                    xx= T.x+floor ( (radius*(cos(theta+alpha)))+.5); //
floor -> to round float to the nearest integer
                                    yy=T.y+floor ( (radius*(sin(theta+alpha)))+.5);


                                if (xx<0)
                                        xx=xx+borderx;


                                if (xx>=borderx)
                                        xx=xx-borderx;


                                if (yy<0)
                                        yy=yy+bordery;


                                if (yy>=bordery)
                                        yy=yy-bordery;

                                temp_layer[xx+yy*imageWidth]=255;
                        }

                        // binfill the temp_layer
                        for(xx=T.x-prtcl_window/2;xx<=T.x+prtcl_window/2;xx++)
                                for                                                     (yy=T.y-
prtcl_window/2;yy<=T.y+prtcl_window/2;yy++)
                                {
                                        i=xx;j=yy;
                                        if (i<0)
                                        i=i+borderx;

                                        if (i>=borderx)
                                                i=i-borderx;


                                        if (j<0)
                                                j=j+bordery;


                                        if (j>=bordery)
                                                j=j-bordery;
```

```
                                        lft=i-1;rht=i+1;
                                        up=j-1;dwn=j+1;
                                        if (lft<0)
                                        lft=lft+borderx;


//                                              if (lft>borderx)
//                                                      lft=lft-borderx;
//                                              if (rht<0)
//                                              rht=rht+borderx;                // lft can
not be > borderx and rht cant be < 0


                                        if (rht>=borderx)
                                                rht=rht-borderx;

                                        if (up<0)
                                                up=up+bordery;
                                        if (dwn>=bordery)
                                                dwn=dwn-bordery;


                                        if      (       (temp_layer[i+j*imageWidth]==0)     &&
(temp_layer[lft+j*imageWidth]==255) && (temp_layer[i+up*imageWidth]==255) &&
                                         (temp_layer[rht+j*imageWidth]==255)            &&
(temp_layer[i+dwn*imageWidth]==255) )
                                         temp_layer[i+j*imageWidth]=255; // binfill.....

                                }
                                //now erase the particle 'ip' from imgdata

                        for(i=T.x-prtcl_window/2;i<=T.x+prtcl_window/2;i++)
                        for (j=T.y-prtcl_window/2;j<=T.y+prtcl_window/2;j++)
                        {

                                        xx=i;yy=j;
                                        if (xx<0)
                                                xx=xx+borderx;


                                        if (xx>=borderx)
                                                xx=xx-borderx;


                                        if (yy<0)
                                                yy=yy+bordery;
```

```
                                        if (yy>=bordery)
                                                yy=yy-bordery;
                                        if (temp_layer[xx+yy*imageWidth]==255)
                                                imgdata[xx+yy*imageWidth]=0;

                        }
                        //and erase the particle from temp_layer....
                        for(xx=T.x-prtcl_window/2;xx<=T.x+prtcl_window/2;xx++)
                                for                                                     (yy=T.y-
prtcl_window/2;yy<=T.y+prtcl_window/2;yy++)
                                {
                                                i=xx;j=yy;



                                                if (i<0)
                                                i=i+borderx;


                                                if (i>=borderx)
                                                        i=i-borderx;


                                                if (j<0)
                                                        j=j+bordery;


                                                if (j>=bordery)
                                                        j=j-bordery;

                                                temp_layer[i+j*imageWidth]=0;
                                }
                                //      writeGrayScaleDataToBmpFile("check33.bmp",
imageWidth, imageHeight, imgdata);
//              writeGrayScaleDataToBmpFile("check11.bmp",              imageWidth,
imageHeight, temp_layer);
                i=3;
                // once the particle is removed .. we need the actual angle for the
calculations
                alpha=part_info[particle_counter].angle; // now use the actual angle for
the calculations
                if (alpha!=0.0)
                        direction=alpha/fabs(alpha); // its the sign of the angle (positive or
negative)
```

```
                else
                        direction=1;
                temp_angle=alpha_rot;
                alpha_rot=alpha_rot+((prtcl[ip].area-
avg_size)/avg_size)*((5*Pi/180)/2)*(temp_angle/1.047197551);
                alpha_rot=alpha_rot+((prtcl[ip].asp_ratio                                -
avg_aspect_ratio)/avg_aspect_ratio)*((5*Pi/180)/2)*(temp_angle/1.047197551);
                alpha_rot=alpha_rot+((fabs(alpha)-
avg_angle)/avg_angle)*((5*Pi/180)/2)*(temp_angle/1.047197551);     //1.047     is     in
radians(60 deg)

                                                        // last term is to make the 'const' a funct. of
alpha_rot

                 if (fabs(alpha)<alpha_rot) // alpha rot is the angle by which every particle
should rotate  and
                                theta_rot =0;       // theta rot is the new angle of the particle
..after rotation by alpha rot
                 else
                                theta_rot=alpha-(direction)*alpha_rot; // before this add
the step where the angle of rotation is adjusted
                                                                        //     for     the     size,     the
angle of orientation and aspect ratio.
                fprintf(angle_file,"%f\t%f\t",alpha*180/Pi,alpha_rot*180/Pi);
                alpha_rot=temp_angle;
                        // particle transfer from 'prtcl' to small_part_data ....
                        for (k=1;k<=prtcl[ip].area;k++)
                        {
                                xx=prtcl[ip].fillx[k]-prtcl[ip].xc+150;
                                yy=prtcl[ip].filly[k]-prtcl[ip].yc+60;
                                small_part_img[xx+yy*300]=255;
                        } //// NOTE : xc and yc for the prtcl in the small_part_img are 150
and 60 respec.....
        //              writeGrayScaleDataToBmpFile("part1.bmp",        300,        120,
small_part_img);
                temp_angle=alpha;

                        alpha=part_info[particle_counter].angle-prtcl[ip].angle;  // alpha
back to relative since we are dealing with the image now
                        theta_rot=theta_rot-prtcl[ip].angle;

                alpha=
rot_angle(imgdata,small_part_img,theta_rot,300,120,150,60,T.x,T.y,imageWidth,directio
n,borderx,bordery,alpha);
                for(k=0;k<300;k++)                                                      //
^^^x_s,y_s,x_c,y_c
```

```
                        for (l=0;l<120;l++)
                                small_part_img[k+l*300]=0;

                        alpha=alpha+prtcl[ip].angle; // again the angle for the calculations
(real particle angle)
                        i=0;
                        for (k=-90;k<=90;k=k+10)
                        {
                                i++;
                                if ((alpha*180/Pi)<=k) break;
                        }
                        if (angle_bin_arr[i][3]>=angle_bin_arr[i][2])    // if after rotating
the angle throws off the bell curve
                        {
                                alpha=temp_angle;  // to get the angle of the particle before
rotation
                                i=0;
                                for (k=-90;k<=90;k=k+10)
                                {
                                        i++;
                                        if ((alpha*180/Pi)<=k) break;
                                }
                                angle_bin_arr[i][3]++;
                                part_info[particle_counter].angle=alpha;        // in this case
the .angle will remain the same as earlier == alpha
                                fprintf (angle_file,"%f\n",alpha*180/Pi);
//                              continue; // do not continue; but place the particle back on
the imgdata with angle alhpa (== temp_angle)
                        }
                        else //added the else brackett and removed continue frm above ;
since the particle has been deleted frm the imgdata, need to put it back with the angle
alpha
                        {
                                angle_bin_arr[i][3]++;
                                part_info[particle_counter].angle=alpha; //    place the new
angle in the part info data
                                fprintf(angle_file,"%f\n",alpha*180/Pi);
                        }//
                // now place the particle on the imgdata via temp_layer with the new angle
'alpha'
                        alpha=alpha-prtcl[ip].angle; // again get the relative angle ;
                        for (i=1;i<=prtcl[ip].area;i++)
                        {
                                xx=prtcl[ip].fillx[i]-prtcl[ip].xc;//+T.x;
                                yy=prtcl[ip].filly[i]-prtcl[ip].yc;//+T.y;
```

```
                        radius=sqrt(xx*xx+yy*yy);
                                if (xx==0)
                                {
                                        theta=Pi/2;
                                        if (yy<0) theta=-1*theta;
                                }
                                else
                                        theta=atan(float(yy)/float(xx));
                                if (xx<0) theta = theta + Pi;

                                xx= T.x+floor ( (radius*(cos(theta+alpha)))+.5); //
floor -> to round float to the nearest integer
                                yy=T.y+floor ( (radius*(sin(theta+alpha)))+.5);


                        if (xx<0)
                                xx=xx+borderx;


                        if (xx>=borderx)
                                xx=xx-borderx;


                        if (yy<0)
                                yy=yy+bordery;


                        if (yy>=bordery)
                                yy=yy-bordery;

                        temp_layer[xx+yy*imageWidth]=255;
                }

                // binfill the temp_layer
                for(xx=T.x-prtcl_window/2;xx<=T.x+prtcl_window/2;xx++)
                        for                                                     (yy=T.y-
prtcl_window/2;yy<=T.y+prtcl_window/2;yy++)
                        {
                                i=xx;j=yy;
                                if (i<0)
                                i=i+borderx;

                                if (i>=borderx)
                                        i=i-borderx;
```

```
if (j<0)
        j=j+bordery;


if (j>=bordery)
        j=j-bordery;

lft=i-1;rht=i+1;
up=j-1;dwn=j+1;
if (lft<0)
lft=lft+borderx;

if (rht>=borderx)
        rht=rht-borderx;

if (up<0)
        up=up+bordery;
if (dwn>=bordery)
        dwn=dwn-bordery;


if ( (temp_layer[i+j*imageWidth]==0) &&
(temp_layer[lft+j*imageWidth]==255) && (temp_layer[i+up*imageWidth]==255) &&
 (temp_layer[rht+j*imageWidth]==255) &&
(temp_layer[i+dwn*imageWidth]==255) )
 temp_layer[i+j*imageWidth]=255; // binfill.....

        }
        //now put the particle 'ip' on imgdata

for(i=T.x-prtcl_window/2;i<=T.x+prtcl_window/2;i++)
for (j=T.y-prtcl_window/2;j<=T.y+prtcl_window/2;j++)
{

        xx=i;yy=j;
        if (xx<0)
                xx=xx+borderx;


        if (xx>=borderx)
                xx=xx-borderx;


        if (yy<0)
```

```
                yy=yy+bordery;


            if (yy>=bordery)
                yy=yy-bordery;

            if (temp_layer[xx+yy*imageWidth]==255)
            {
                if
(imgdata[xx+yy*imageWidth]==255)
                {
                    k=3;

                    k=3;

                }

                imgdata[xx+yy*imageWidth]=255;
            }

        }
        //erase the particle from temp_layer
        for(xx=T.x-prtcl_window/2;xx<=T.x+prtcl_window/2;xx++)
            for                                                    (yy=T.y-
prtcl_window/2;yy<=T.y+prtcl_window/2;yy++)
            {
                i=xx;j=yy;



                if (i<0)
                i=i+borderx;


                if (i>=borderx)
                    i=i-borderx;


                if (j<0)
                    j=j+bordery;


                if (j>=bordery)
                    j=j-bordery;

                temp_layer[i+j*imageWidth]=0;
```

```
                                }
//                              writeGrayScaleDataToBmpFile("check44.bmp",
imageWidth, imageHeight, imgdata);
                                xx=3;




                } //
                i=0;
                for (k=-90;k<=90;k=k+10)
                {
                        i++;

        fprintf(ang_bin_arr,"%d\t%d\t%d\n",angle_bin_arr[i][1],angle_bin_arr[i][2],angle
_bin_arr[i][3]);
                }
                fclose(ang_bin_arr);

                        itoa (fno,n1,10);
            strcpy (str,imagename);
                if (no_sims>1) { strcat (str,"-"); strcat (str,n1); }
                alpha_rot=alpha_rot*180/Pi;
                itoa (alpha_rot,n1,10);
                strcat (str,"-rot-");strcat (str,n1);
                strcat (str,".bmp");
                writeGrayScaleDataToBmpFile(str, imageWidth, imageHeight, imgdata);
                alpha_rot=alpha_rot*Pi/180;
//                              writeGrayScaleDataToBmpFile("TiB_DOE_E-2b.bmp",
imageWidth, imageHeight, imgdata);
                        fclose(angle_file);
//              writeGrayScaleDataToBmpFile("check11.bmp",              imageWidth,
imageHeight, temp_layer);
                i=3;
                } // end of loop for the rot_counter // all the rotations applied on the image
          } // end of "if (alpha_rot !=0) "
                i=3;




                ang_bin_arr=fopen("prt_bin_output.txt","w");
                for(i=1;i<no_of_bin;i++)
                {

        fprintf(ang_bin_arr,"%d\t%d\t%d\t%d\n",prt_size_bin[i][0],prt_size_bin[i][1],prt
_size_bin[i][2],prt_size_bin[i][3]);
```

```
		}
		fclose(ang_bin_arr);

//////////////////////////////// PARTICLE ROTATION FINISHED
////////////////////////////////////////////

//////////////////////////  lineal path  ////////////////////////////////////////////////////////

	for (i=0; i<4; i++)
		for (j=0; j<=end; j++)
			p[i][j]=0;


		for(r=start;r<end;r=r+unit_r)
			q[r]=0;
	imageWidth=x_s;imageHeight=y_s;

		for(ya=0 ;ya<y_s ;ya=ya+unit_xy)
		{
			pa=(imgdata[0+ya*imageWidth]==255)?(0):(1);  // value at (0,ya)
			yb=ya;
			len_cnt=1;
			for(xa=1;xa<x_s;xa=xa+unit_xy)
			{
				pb=(imgdata[xa+yb*imageWidth]==255)?(0):(1);
//xb is xa in this case
				if (pa==pb)
				{
					len_cnt++;
				}
				else
				{
					counter=len_cnt;
					if (len_cnt>end) counter=end;
					for ( i=1;i<=counter;i++)
						p[pa][i]=p[pa][i]+len_cnt-i+1;

					pa=pb;
					len_cnt=1;
				}
			}


			counter=len_cnt;
			if (len_cnt>end) counter=end;
```

```
                    for (i=1;i<=counter;i++)
                            p[pa][i]=p[pa][i]+len_cnt-i+1;  // this is to account for the last piece of line segment in the x direction..

                }


                                    // to calculate the total number of line segmets that can be used i.e. q[r]

                for (i=1;i<=end;i++)
                {
                        q[i]=(x_s-i+1)*y_s;
                        len_cnt=len_cnt+1;
                }


   toterr=0;
        maxerr=0;
        fprintf(lpath,"output_x");
        fprintf(lpath, "\n r");


        cnt=1;
        for (r=start; r<end; r=r+unit_r)
        {
                fprintf(lpath, "\n%d\t", r-1);
                for (i=0;i<1;i++)
                {
                        error[r-1]=((lpreal[r-1]-float(p[i][r])/q[r]))*100/lpreal[r-1];
//lpreal[r-1] coz the array starts with '0' and r with 1
                        if (error[r-1]<0) error[r-1]=-1*error[r-1];
                        fprintf(lpath, "%f\t%f", float(p[i][r])/q[r],error[r-1]);
//                      avg2pt[cnt][0][i]=avg2pt[cnt][0][i]+float(p[i][r])/q[r];
                        cnt++;
                        if (error[r-1]>maxerr) {maxerr=error[r-1];worstr=r-1;}
                        toterr=toterr+error[r-1];
                }
        }
   avgerr=toterr/(cnt-1);
        fclose(lpath);
```

```
errfile=fopen("error-file.txt","a");
fprintf(errfile,"%f\t%f  %d\n",avgerr,maxerr,worstr );
fclose(errfile);
cnt=no_sims;
if (no_sims>10) cnt=10;
for (i=1;i<=cnt;i++)  // changed from no_sims to cnt
{
        if (maxerr<=err[i].mxerr)
        {
                for (r=cnt-1;r>=i;r=r-1) // changed from no_sims to cnt
                {
                        err[r+1].fn=err[r].fn;
                        err[r+1].mxerr=err[r].mxerr;
                        err[r+1].agerr=err[r].agerr;
                }
                err[i].fn=fno;
                err[i].mxerr=maxerr;
                err[i].agerr=avgerr;
                break;
        }
}



strcpy (str3,str);
//strcat (str3,".txt");
//fp=fopen (str2, "a");
strcat (str3,".bmp");
//strcat (str,1);
//writeGrayScaleDataToBmpFile(str3, imageWidth, imageHeight, imgdata);
i=10;




//return (*result);
        for (i=0;i<=imageHeight;i++)
                for(j=0;j<=imageWidth;j++)
                        imgdata[j+i*imageWidth]=0;

fp=fopen ("check123.txt","w");
for (i=1;i<11;i++)

fprintf(fp," %d \n",ovrlpd_prt[i].xc);
```

```
// fno++;  debug 503 error
//delete [] px_prt_info;
//delete [] ovrlpd_prt;

delete  px_prt_info;
delete  ovrlpd_prt;
for (i=1;i<11;i++)

fprintf(fp," %d \n",ovrlpd_prt[i].xc);

fclose(fp);


} //end of the while fno loop ..now four for loops

errfile=fopen("error-file.txt","a");
fprintf(errfile, "  ***********    best matched simulations ***********\n");
for (i=1;i<=min(10,no_sims);i++)
fprintf(errfile,"%d\t%d\t%d\t%0.3f\t
%0.3f\t\t%f\t%f\n",err[i].fn,pmt[err[i].fn].x,pmt[err[i].fn].y,pmt[err[i].fn].Aa,pmt[err[i].f
n].lwr_lmt,err[i].agerr,err[i].mxerr);
fprintf(errfile,"overlap ratio = %0.3f\tvolume frac = %0.3f ang =
%d",overlap_ratio,vol_frac,ang);
fclose(errfile);

 }


void sort1( float arr[], int arr2[],int left, int right)
{
        int i,last;
        void swap1(float arr[],int i ,int j);
        void swap(int arr[],int i,int j);
        if (left>=right)
                return;
        swap1(arr,left,(left+right)/2);
        swap(arr2,left,(left+right)/2);


        last =left;
        for (i=left+1; i<=right;i++)
                if (arr[i]<arr[left])
                {

                        swap1(arr, ++last,i);
                        swap(arr2, last,i);
```

```
                }



                swap1(arr,left,last);
                swap(arr2,left,last);

                sort1(arr,arr2,left,last-1);
                sort1(arr,arr2,left+1,right);
}
void swap1(float arr[],int i,int j)
{
        float temp;

        temp =arr[i];
        arr[i]=arr[j];
        arr[j]=temp;
}

float random(int input)

{
        long SimID;
        SimID = (unsigned)time( NULL );
        seedMT(SimID);
        return(input*random53());
}




// function for aspect_ratio

float  asp_rat(unsigned char * big_part_img, int x_s, int y_s)
{
        struct point
        {
                int xx;
                int yy;
        };
        FILE *ferret_data,*ferret_angles;
        point p1,p2;
```

```
	float
aspect_ratio,ferret_min,ferret_max,theta,dist,gamma,ferret_min_angle,ferret_max_angle,
ferret_ang; // theta is in radians here
	int x,y,c1,c2; // c1 and c2 are the intercepts (c1 is intercept on left vertical side
and c2 is on the right verticle side)
	int out;
	x=35;y=90;
	out=0;
	ferret_max=0;ferret_min=x_s; // initializing the ferrets with extereme values
//	c1=0;c2=y_s; // y_s is the y dimension of the image
	float inter_y1,inter_y2;//check;
	float fr1,fr2,avg1,avg2,diff1,diff2;
	ferret_data=fopen("ferret_data.txt","a"); // "w" to "a"

	c1=-x_s; // we start with a given theta and intercept.. keep changing the intercept
till u find the particle (for that given theta)
	c2=x_s+y_s;
//	theta=0;c1=0;c2=y_s;
	theta = -45*Pi/180; // going from -45 to +45 degrees....
	while (theta<=45*Pi/180)
	{
					out=0;
					while (out==0)
					{

						x=0; //y=c1;
						if (c1>y_s) x=floor ((c1-y_s)/tan(theta)+.5);
// see notes
						if (c1<0) x=floor ((c1)/tan(theta)+.5); // see
notes
						y=c1- (floor(x*tan(theta)+.5)); // so y will be
= y_s for c1>y_s and = 0 for c1>0 and otherwise y=c1
						while ((x<=x_s) && (y<=y_s) && (x>=0)
&& (y>=0) )
						{
							if (big_part_img[x+y*x_s]==255)
							{
								p1.xx=x;
								p1.yy=y;
								out=1;
								break;
							}
							x=x+2; //scanning the line from left
to right
							y=c1- (floor(x*tan(theta)+.5));
						}
```

```
                                        if (out==0) c1=c1+1;// increasing the
intercept.. lowering the line 1
                                }
                                out=0;
                                while (out==0)
                                {

                                        x=x_s;//y=c2;
                                        if (c2<0) x=x_s-floor( ((-1*c2)/tan(theta)) +
.5); // check for the -1 sign here
                                        if (c2>y_s) x= x_s + (floor (c2-
y_s)/tan(theta) + .5);
                                        y=floor ((x_s-x)*tan(theta)+.5) +c2;
                                        while ((x>=0) && (y<=y_s) && (x<=x_s)
&& (y>=0) )
                                        {
                                                if (big_part_img[x+y*x_s]==255)
                                                {
                                                        p2.xx=x;
                                                        p2.yy=y;
                                                        out=1;
                                                        break;
                                                }
                                                x=x-2; // scanning the line from right
to left
                                                y=floor ((x_s-x)*tan(theta)+.5) +c2;
                                        }
                                        if (out==0) c2=c2-1;// decreasing the
intercept.. moving  the line 2 UP
                                }
                                inter_y1=p1.yy+p1.xx*tan(theta); // we have c= y
(+) x(tan(theta)) coz of the axis of the image
                                inter_y2=p2.yy+p2.xx*tan(theta);        //fprintf
(ferret_data,"%f        %f      \n",inter_y1,inter_y2);
                                dist=( (inter_y2)- (inter_y1) )*cos(theta);
                                if (dist<0) dist=-1*dist;
                                if      (dist>ferret_max)       {ferret_max=dist;
ferret_max_angle=theta*180/Pi;}
                                if      (dist<ferret_min)       {ferret_min=dist;
ferret_min_angle=theta*180/Pi;} // added the ferret angle part
                                theta=theta + 1*Pi/180;
                        //      fprintf (ferret_data,"%f        %f      %f
        %f      %f\n", dist,ferret_min,ferret_max,  ferret_min_angle,ferret_max_angle-
90);//p2.xx,p2.yy,dist,inter_y2);

        }
```

```
                // theta = 45 to 135 deg.

        c1=c1-y_s; // this will give the intercept on the x axis.....
        c2=x_s+c2; // and we need to go down on Y for c1 and up on Y for c2 ( which is opposite from the X)
        while (theta<=135*Pi/180)
        {
                                        out=0;gamma=theta*180/Pi;
                                        while (out==0)
                                        {

                                                y=y_s;//x=0; //y=c1;
                                                if (c1<0) y=y_s - floor(-1*c1*tan(theta)+.5);     //if (c1>y_s) x=floor ((c1-y_s)/tan(theta)+.5); // see notes
                                                if (c1>x_s) y=y_s - floor(-1*c1*tan(theta) + .5);       //if (c1<0) x=floor ((c1)/tan(theta)+.5); // see notes
                                                x=c1+  floor( (y_s-y)/tan(theta) + .5);//y=c1-(floor(x*tan(theta)+.5)); // so y will be = y_s for c1>y_s and = 0 for c1>0 and otherwise y=c1
                                                while ((x<=x_s) && (y<=y_s) && (x>=0) && (y>=0) )
                                                {
                                                        if (big_part_img[x+y*x_s]==255)
                                                        {
                                                                p1.xx=x;
                                                                p1.yy=y;
                                                                out=1;
                                                                break;
                                                        }
                                                        y=y-2;
                                                //x=x+2;        //scanning the line from left to right
                                                        x=c1+    floor( (y_s-y)/tan(theta) + .5); // scanning the line from bottom to top          //y=c1- (floor(x*tan(theta)+.5));
                                                }
                                                if (out==0) c1=c1+1;// increasing the intercept.. lowering the line 1
                                        }
                                        out=0;
                                        while (out==0)
                                        {

                                                y=0;//x=x_s;//y=c2;
                                                if (c2>x_s) y= floor ((c2-x_s)*tan(theta) + .5); //if (c2<0) x=x_s-floor( ((-1*c2)/tan(theta)) + .5); // check for the -1 sign here
```

```
                                        if (c2<0)    y= floor (c2*tan(theta) + .5);//if (c2>y_s) x= x_s + (floor (c2-y_s)/tan(theta) + .5);
                                        x=c2-  floor(y/tan(theta)+.5);//y=floor ((x_s-x)*tan(theta)+.5) +c2;
                                        while ((x>=0) && (y<=y_s) && (x<=x_s) && (y>=0) )
                                        {
                                                if (big_part_img[x+y*x_s]==255)
                                                {
                                                        p2.xx=x;
                                                        p2.yy=y;
                                                        out=1;
                                                        break;
                                                }
                                                y=y+2;//x=x-2; // scanning the line from right to left
                                                x=c2-           floor(y/tan(theta)+.5); //y=floor ((x_s-x)*tan(theta)+.5) +c2;
                                        }
                                        if (out==0) c2=c2-1;// decreasing the intercept.. moving  the line 2 UP
                                } // axis of the image is not the normal XY axis
                                inter_y1=p1.yy+p1.xx*tan(theta); // we have c= y (+) x(tan(theta)) coz of the axis of the image
                                inter_y2=p2.yy+p2.xx*tan(theta);        //fprintf (ferret_data,"%f      %f      \n",inter_y1,inter_y2);
                                dist=( (inter_y2)- (inter_y1) )*cos(theta);
                                if (dist<0) dist=-1*dist;
                                if      (dist>ferret_max)       {ferret_max=dist; ferret_max_angle=theta*180/Pi;}
                                if      (dist<ferret_min)       {ferret_min=dist; ferret_min_angle=theta*180/Pi;} // added the ferret angle part
                                theta=theta + 1*Pi/180;
                        //      fprintf (ferret_data,"%f      %f      %f      %f      %f\n", dist,ferret_min,ferret_max,ferret_min_angle,ferret_max_angle);//p2.xx,p2.yy,dist,inter_y2);

        } //fprintf(ferret_data," \n\tfinal  \n");
        fprintf (ferret_data,"%f      %f      %f      %f\n", ferret_min,ferret_max,ferret_min_angle,ferret_max_angle-90);
        fclose(ferret_data);
        aspect_ratio=ferret_max/ferret_min;

        fr1=ferret_min_angle;  // fr1 in (-45 to 135)
        if (fr1<0) fr1=180+fr1; // fr1 in (0 to 180)
```

```
	fr2=ferret_max_angle-90; // fr2 in (-135 to 45)
	if (fr2<0) fr2=180+fr2;  // fr2 in (0 to 180)


	avg1=(fr1+fr2)/2;	//avg1 in (0 to 180)
	avg2=avg1+90;			//avg2 in (90 to 240)
	if (avg2>=180) avg2=avg2-180; //avg2 in (0 to 180)

	if (fabs(avg1-fr1)<fabs(avg1-fr2))
		diff1=fabs(avg1-fr1);
	else
		diff1=fabs(avg1-fr2);
	if (fabs(avg2-fr1)<fabs(avg2-fr2))
		diff2=fabs(avg2-fr1);
	else
		diff2=fabs(avg2-fr2);
	if (diff1<diff2)
		ferret_ang=avg1;
	else
		ferret_ang=avg2;	//ferret_ang in (0 to 180)




	if (ferret_ang>90) ferret_ang=-(180-ferret_ang); // we need the final angs from -90 to +90
	ferret_ang=ferret_ang*Pi/180;
	ferret_angles=fopen("ferret_angles.txt","a"); // need to change here for each different input file
	fprintf  (ferret_angles,"%f\n",ferret_ang);
	fclose(ferret_angles);
	dist=3;
	return (aspect_ratio);
}
// rot_angle -> to find the angle at which particle is to be rotated ..
float rot_angle (unsigned char * imgdata, unsigned char * big_part_img, float alpha,int x_s,int y_s, int x_c, int y_c,int Tx, int Ty, int imageWidth,int direction, int borderx, int bordery, float alpha_init) // alpha initial // start frm alpha_init
{

	//		int writeGrayScaleDataToBmpFile(char * , unsigned int , unsigned int , unsigned char * );
```

```
//BITMAPINFOHEADER bitmapInfoHeader;
//unsigned char *bitmapData;
        int finish,i,j,xx,yy;
        float radius,theta,temp_ang;
        finish=0;
        alpha_init=alpha_init-direction*2*Pi/180;
        while (finish==0)
        {


                for(i=0;i<x_s;i++)
                {
                        for (j=0;j<y_s;j++)
                         {
                                if (big_part_img[i+j*x_s]==255)
                                {

                                        xx=i-x_c;
                                        yy=j-y_c;
                                       // if ((yy/xx)<0)
                                       //         theta=1;
                                        radius=sqrt(xx*xx+yy*yy);
                                        if (xx==0)
                                        {
                                                theta=Pi/2;
                                                if (yy<0) theta=-1*theta;
                                        }
                                        else
                                                theta=atan(float(yy)/float(xx));
                                        if (xx<0) theta = theta + Pi;

                                        xx=                floor                (
(radius*(cos(theta+alpha_init)))+.5); // floor -> to round float to the nearest integer
                                        yy=floor                                (
(radius*(sin(theta+alpha_init)))+.5);     // make xx = xx +tx and then check for xx<0 or
>borderx
                                        xx=Tx+xx;
                                        yy=Ty+yy;
                                        if (xx<0)
                                                xx=xx+borderx;


                                       if (xx>=borderx)
                                                xx=xx-borderx;
```

```
                                        if (yy<0)
                                                yy=yy+bordery;


                                        if (yy>=bordery)
                                                yy=yy-bordery;

                                         if  (imgdata[(xx)+(yy)*imageWidth]==255)
//if (imgdata[(Tx+xx)+(Ty+yy)*imageWidth]==255)
                                        {
                                                finish=1;
                                        //      alpha_init=alpha_init-
direction*2*Pi/180; // move the alpha other way around
                                                break;
                                        }
                                }
                        }
                        if (finish==1) break;
                } // finish == 1 -> overlap
                  // finish == 0 -> no overlap
                if (finish==1)
                {
                        if (alpha==alpha_init)
                                alpha_init=temp_ang;
                        else
                                alpha_init=alpha_init+direction*2*Pi/180;
                        break;
                }
                if (finish == 0)
                {
                        if (alpha==alpha_init)
                        {
                                finish=1;
                                break;
                        }
                        if (fabs(alpha_init-alpha)>2*Pi/180)
                        {
                                alpha_init=alpha_init-direction*2*Pi/180;
                                temp_ang=alpha_init;
                        }
                        else
                                alpha_init=alpha;
                }
                i=fabs(alpha_init-alpha);
        /*              if (fabs(alpha)>fabs(alpha_init))  // alpha is the final intended angle
```

```
                        {                                                                   //
alpha_init is the intial angle
                                finish=1;
                                alpha=alpha_init;
                        }

                        if (finish==1) break;   */

        }


finish=3;
return (alpha_init);




}
```

// Two Point Probability code

```
# define Im_in(x,y) (*(buf_in+((y-1)*x_s+x-1)))
# define Im_in1(x,y) (*(buf_in1+((y-1)*x_s+x-1)))
# define Im_in2(x,y) (*(buf_in2+((y-1)*x_s+x-1)))
# define Im_out(x,y) *(buf_out+((y-1)*x_s+x-1))
# define Clip(x) ( (x) > (255) ? (255): (x))
# define Clip_min(x) ( (x) < (0) ? (0): (x))
# define min(x,y) ( (x)>(y)?(y):(x) )
# define max(x,y) ( (x)>(y)?(x):(y) )
#include <stdlib.h>
# include "math.h"
# include "stdio.h"
#include <time.h>
#include <stdio.h>
#include <limits.h>
#include <math.h>
#include <iostream.h>
#include <windows.h>


#define Pi 3.14159265358979

#define in_one_cell 20

#include <string.h>
```

```
//two point


// check for the image .... it reads the image upside down.....!!!!!
unsigned char *LoadBitmapFile(char *filename, BITMAPINFOHEADER
*bitmapInfoHeader)
{

        FILE *filePtr; //our file pointer
        BITMAPFILEHEADER bitmapFileHeader; //our bitmap file header
        unsigned char *bitmapImage; //store image data
        int imageIdx=0; //image index counter
        unsigned char tempRGB; //our swap variable

        //open filename in read binary mode
        filePtr = fopen(filename,"rb");
        if (filePtr == NULL)
        {       printf(" where is the file ?\n ");
                return NULL;
        }

        //read the bitmap file header
        fread(&bitmapFileHeader, sizeof(BITMAPFILEHEADER),1,filePtr);

        //verify that this is a bmp file by check bitmap id
        if (bitmapFileHeader.bfType !=0x4D42)
        {
                fclose(filePtr);
                return NULL;
        }

        //read the bitmap info header
        fread(bitmapInfoHeader, sizeof(BITMAPINFOHEADER),1,filePtr);

        //move file point to the begging of bitmap data
        fseek(filePtr, bitmapFileHeader.bfOffBits, SEEK_SET);

        //allocate enough memory for the bitmap image data
        bitmapImage = (unsigned char*)malloc(bitmapInfoHeader->biSizeImage);

        //verify memory allocation
        if (!bitmapImage)
        {
                free(bitmapImage);
                fclose(filePtr);
                return NULL;
```

```
	}

	//read in the bitmap image data
	fread(bitmapImage,bitmapInfoHeader->biSizeImage,1,filePtr);

	//make sure bitmap image data was read
	if (bitmapImage == NULL)
	{
		fclose(filePtr);
		return NULL;
	}

/*	//swap the r and b values to get RGB (bitmap is BGR)
	for (imageIdx = 0;imageIdx < bitmapInfoHeader->biSizeImage;imageIdx+=3)
	{
		tempRGB = bitmapImage[imageIdx];
		bitmapImage[imageIdx] = bitmapImage[imageIdx + 2];
		bitmapImage[imageIdx + 2] = tempRGB;
	}*/

		//close file and return bitmap iamge data
		fclose(filePtr);
		return bitmapImage;
}
int _stdcall C_Add_Sub_Mul(unsigned char Oper , unsigned char Opt, unsigned char *buf_in1,unsigned char *buf_in2, unsigned char *buf_out, int x_s, int y_s, unsigned char *debug_ar)
{
	unsigned char *imgdata;
	BITMAPINFOHEADER bitmapInfoHeader;
	int M;   // changed M from const to int
	int i,j,k,imageWidth,imageHeight;
	char str[40],n1[10],str2[40];
//	imageWidth=1900;imageHeight=1450;
		imageWidth=x_s;
	imageHeight=y_s;
//	imgdata=(unsigned char *) malloc ( (imageHeight+2)*(imageWidth+2)*sizeof(unsigned char));
	imgdata = new unsigned char[(imageHeight+2)*(imageWidth+2)];
	i=10;
	int xa,xb,ya,yb,r,cnt,no_images,start,len_cnt,counter ;
	int dir_x=1;int dir_y=11; int dir_45=11;int frame_len;
	start=0;
	const end=500;//const end=1000;
	M=end;
```

```
        float lut[end];
        unsigned long p[4][end+2],q[end+2];
//      double sqr[M+1][M+1];

        int pa,pb,pd,unit_r; //unit_r is the r=r+ ??

        int unit_xy,out1;
        i=10;
        FILE *fp,*fpx,*fpy,*fp45;
        float avg2pt[2000][3][3]; // 1st dim. is the value at any given r ,...
//                                                              2nd dim. is the for direction
.. 0 = x . 1 = y , 2 = 45
//                                                              3rd dim. is for
particles(=0) and matrix(=1) values

        for (i=start-1;i<=end;i++)
                lut[i]=i*sqrt(2);
        for (i=0;i<2000;i++)
                for (j=0;j<=3;j++)
                        for (k=0;k<=1;k++)
                        avg2pt[i][j][k]=0;

        unit_r=3;// earlier 26
int no_of_images;               no_of_images=0;   // this is the actual number of images
used (start with zero)




        unit_xy=5;  // its the unit for x and y grid ... earlier used 10  / 9/18/07


                no_images=5; // image no. of the last image ....



        for (k=1;k<=no_images;k++) { // this is the start of the image loop

                no_of_images++;
                itoa (k,n1,10);


                strcpy (str,"TiB_DOE_I");

                strcat (str,n1);
                        strcat (str,"-rot-0");
```

```
        strcpy (str2,str);
        strcat (str,".bmp");
        imgdata = LoadBitmapFile(str,&bitmapInfoHeader);

        strcpy (str,str2);
        strcat (str,"-2 point-");
        strcat (str,n1);
        strcat (str,".txt");

//      imageWidth=5000;imageHeight=5000;
        fp=fopen(str,"w");

//      for (i=0; i<=M; i++)
//              for (j=0; j<=M; j++)
//                      sqr[i][j]=sqrt(i*i+j*j);

        for (i=0; i<4; i++)
                for (j=0; j<=M; j++)
                        p[i][j]=0;


                if (dir_x==1){
                for(r=start;r<end;r=r+unit_r)
                        q[r]=0;
//      x_s=5000;y_s=5000;
//      for(r=start;r<end;r=r+unit_r)
        imageWidth=x_s;imageHeight=y_s;



                for (i=0; i<4; i++)
                        for (j=0; j<M; j++)
                                p[i][j]=0;


                for(r=start;r<end;r=r+unit_r)
                {
                        for(xa=r;xa<x_s-r;xa=xa+unit_xy)
                        {
                                for(ya=0 ;ya<y_s ;ya=ya+unit_xy)
                                {
                                        pa=(imgdata[xa+ya*imageWidth]==255)?(0):(1);
                                        xb=xa-r;
                                        yb=ya;//-int(r/ang);
                                        pb=(imgdata[xb+yb*imageWidth]==255)?(0):(1);
```

```
                                        p[pa+pa+pb][r]++;
                                        xb=xa+r;
                                        yb=ya;//+int(r/ang);
                                        pb=(imgdata[xb+yb*imageWidth]==255)?(0):(1);
                                        p[pa+pa+pb][r]++;
                                }
                        }
                }


                for (r=start; r<end; r=r+unit_r)
                        q[r]=0;

                for (r=start; r<end; r=r+unit_r)
                        for (pd=0; pd<4; pd++)
                                q[r]=q[r]+p[pd][r];




        fprintf(fp,"output_x");
        fprintf(fp, "\n r");
/*      for (pd=0; pd<4; pd++)
                fprintf(fp,"\tP%i%i",div(pd,2).quot,div(pd,2).rem);*/

        cnt=0;
        for (r=start; r<end; r=r+unit_r)
        {
                fprintf(fp, "\n%d\t", r);
                for (i=0;i<=0;i++)
                {
                        fprintf(fp, "%f\t", float(p[i][r])/q[r]);
                        avg2pt[cnt][0][i]=avg2pt[cnt][0][i]+float(p[i][r])/q[r];
                        cnt++;
                }
        }

//      fclose(fp);
                } // end of "if "
        for (i=0; i<4; i++)
                for (j=0; j<M; j++)  // p[i][0] will be same for x and y direction
                        p[i][j]=0;
```

```
// ---------------------- x done y starts ------------------------------------------------

                if (dir_y==1){
                        for (r=start; r<end; r=r+unit_r)
                q[r]=0;

                for(r=start;r<end;r=r+unit_r)
                        for(xa=0 ;xa<x_s ;xa=xa+unit_xy)
                                for(ya=r;ya<y_s-r;ya=ya+unit_xy)
                                {
                                        pa=(imgdata[xa+ya*imageWidth]==255)?(0):(1);
                                        xb=xa;//-int(r/ang);
                                        yb=ya-r;
                                        pb=(imgdata[xb+yb*imageWidth]==255)?(0):(1);
                                        p[pa+pa+pb][r]++;
                                        xb=xa;//+int(r/ang);
                                        yb=ya+r;
                                        pb=(imgdata[xb+yb*imageWidth]==255)?(0):(1);
                                        p[pa+pa+pb][r]++;
                                }



//      fp=fopen ("output_y.txt", "w");
                        fprintf(fp,"\noutput_y");


                for (r=start; r<end; r=r+unit_r)
                        q[r]=0;

                for (r=start; r<end; r=r+unit_r)
                        for (pd=0; pd<4; pd++)
                                q[r]=q[r]+p[pd][r];

        fprintf(fp, "\n r");
/*      for (pd=0; pd<4; pd++)
                fprintf(fp,"\tP%i%i",div(pd,2).quot,div(pd,2).rem);*/

        cnt=0;
        for (r=start; r<end; r=r+unit_r)
        {
                fprintf(fp, "\n%d\t", r); // changed from r-1 to r
                for (i=0;i<=0;i++)
                {
```

```
                        fprintf(fp, "%f\t", float(p[i][r])/q[r]);
                        avg2pt[cnt][1][i]=avg2pt[cnt][1][i]+float(p[i][r])/q[r];
                        cnt++;
                }
        }
                }




                        for (i=0; i<4; i++)
                for (j=0; j<M; j++)  // p[i][0] will be same for x and y direction
                        p[i][j]=0;
//                      fclose(fp);
// ----------------------------------------------------------y done --- 45 starts  -----------------

int x_len,y_len;

                for (r=start; r<end; r=r+unit_r)
                        q[r]=0;

                for(r=start;r<end;r=r+unit_r)
                        for(xa=r ;xa<x_s-r ;xa=xa+unit_xy)
                                for(ya=r;ya<y_s-r;ya=ya+unit_xy)
                                {

                                        pa=(imgdata[xa+ya*imageWidth]==255)?(0):(1);
                                        xb=xa-r;
                                        yb=ya-r;//+int(r/ang);
                                        pb=(imgdata[xb+yb*imageWidth]==255)?(0):(1);
                                        p[pa+pa+pb][r]++;
                                        xb=xa+r;
                                        yb=ya+r;//-int(r/ang);
                                        pb=(imgdata[xb+yb*imageWidth]==255)?(0):(1);
                                        p[pa+pa+pb][r]++;

                                }


        fprintf(fp,"\noutput_45");


                for (r=start; r<end; r=r+unit_r)
                        for (pd=0; pd<4; pd++)
                                q[r]=q[r]+p[pd][r];
```

```
        fprintf(fp, "\n r");
/*      for (pd=0; pd<4; pd++)
                fprintf(fp,"\tP%i%i",div(pd,2).quot,div(pd,2).rem);*/

        cnt=0;
        for (r=start; r<end; r=r+unit_r)
        {
                fprintf(fp, "\n%f\t", lut[r]);
                for (i=0;i<=0;i++)
                {
                        fprintf(fp, "%f\t", float(p[i][r])/q[r]);
                        avg2pt[cnt][2][i]=avg2pt[cnt][2][i]+float(p[i][r])/q[r];
                        cnt++;
                }
        }



        }

        fclose(fp);
}




        if (no_of_images>1)
        {
                strcpy (str,str2);
                strcat (str,"-2 point-avg-x.txt");
                fpx=fopen(str,"w");
                        strcpy (str,str2);
                strcat (str,"-2 point-avg-y.txt");
                fpy=fopen(str,"w");
                        strcpy (str,str2);
                strcat (str,"-2 point-avg-45.txt");
                fp45=fopen(str,"w");


                        cnt=0;
                for (r=start; r<end; r=r+unit_r)
                {
                        fprintf(fpx,"\n%d",r);
                        fprintf(fpy,"\n%d",r);
```

```
                    fprintf(fp45,"\n%f",lut[r]);
                    for (i=0;i<=0;i++)
                    {

                        fprintf(fpx,"\t%f",(avg2pt[cnt][0][i])/no_of_images);
                        fprintf(fpy,"\t%f",(avg2pt[cnt][1][i])/no_of_images);
                        fprintf(fp45,"\t%f",(avg2pt[cnt][2][i])/no_of_images);
                        cnt++;
                    }
                }

                fclose(fpx);fclose(fpy);fclose(fp45);
        }



return (101);
}
```

# REFERENCES


[1] Slipenyuk, A., Kuprin V., Milman Y., Spowart J. E.and Miracle D. B., "The effect of matrix to reinforcement particle size ration (PSR) on the microstructure and mechanical properties of a P/M processed AlCuMn/SiCp MMC," *Materials Science and Engineering*, vol. A381, pp. 165-170, 2004.

[2] Yang, N., Boselli J.and Sinclair I., "Simulation and quantitative assessment of homogeneous and inhomogeneous particle distributions in particulate metal matrix composites," *Journal of Microscopy*, vol. 201, pp. 189-200, 2001.

[3] Saylor, D. M., Fridy J., El-Dasher B. S., Jung K.-Y.and Rollett A. D., "Statistically representative three-dimensional microstructure based on orthogonal observation sections," *Metallurgical and Materials Transactions*, vol. 35A, pp. 1969, 2004.

[4] Surappa, M. K., "Aluminum matrix composite: Challenges and opportunities," *Sadhana*, vol. 28, pp. 319-334, 2003.

[5] www.dwa-dra.com

[6] Geiger, A. L.and Walker J. A., "Processing and properties of discontinuously reinforced aluminum composites," *JOM*, vol. 43, pp. 8-15, 1991.

[7] Isalak, A. Ferrous Powder Metallurgy. Cambridge: Cambridge International Science Publishing, 1995.

[8] Fard, R. R.and Akhlaghi F., "Effect of extrusion temperature on the microstructure and porosity of A356-SiCp composites," *Journal of Materials Processing Technology*, vol. 187, pp. 433-436, 2007.

[9] Prasad, V. V. B., Bhat B. V. R., Mahajan Y. R.and Ramakrishnan P., "Effect of extrusion parameters on structure and properties of 2124 aluminum alloy matrix composites," *Materials and Manufacturing Processes*, vol. 16, pp. 841-853, 2001.

[10] Bruno, G., Girardin E., Giuliani A., Manescu m., Rustichelli F., O'Donnel B.and Hugh P. E. M., "Residual stress analysis in aerospace MMC materials by neutron diffraction," *Applied Physics A*, vol. 74, pp. 2002.

[11] Rawal, S., "Metal-Matrix Composites for Space Applications," *JOM*, vol. 53, pp. 14-17, 2001.

[12] Pandey, A. B., Majumdar B. S.and Miracle D. B., "Effect of aluminum particles on the fracture toughness of a 7093/SiC/15p composite," *Materials Science and Engineering*, vol. A259, pp. 296, 1999.

[13] Lu, Y. X., Lee C. S., Meng X. M.and Li R. K. Y., "Effect of matrix strength on the fracture mechanism of SiC particle reinforced Al-Cu matrix composites under dynamic loading," *Journal of Materials Science Letters*, vol. 18, pp. 533-535, 1999.

[14] Caley, W. F., Paton B., Bishop D. P.and Kipouros G. J., "On enhancing the interfacial chemisty of a simulated AA2014-SiCp composite material," *Journal of Materials Science*, vol. 38, pp. 1755-1763, 2003.

[15] Hong, S.-J., Kim H.-M., Huh D., Suryanarayana C.and Chun B. S., "Effect of clustering on the mechanical properties of SiC particulate-reinforced aluminum alloy 2024 metal matrix composites," *Materials Science and Engineering*, vol. A347, pp. 198-204, 2003.

[16] Boselli, J., Pitcher P. D., Gregson P. J.and Sinclair I., "Quantitative assesment of paritcle distribution effects on short crack growth in SiCp reinforced Al-alloys," *Scripta Materialia*, vol. 38, pp. 839-844, 1998.

[17] Bindumadhavan, P. N., Chia T. K., Chanrasekaran M., Wah H. K., Lam L. N.and Prabhakar O., "Effect of particle porosity clusters on the tribological behabior of cast aluminum alloy A356-SiCp metal matrix composites," *Materials Science and Engineering*, vol. A315, pp. 217-226, 2001.

[18] Deng, X.and Chawla N., "Modeling the effect of particle clustering on the mechanical behavior of SiC particle reinforced Al matrix composites," *Journal of Materials Science*, vol. 41, pp. 5731-5734, 2006.

[19] Froes, E. H., "How to Market Titanium: Lower the Cost," *JOM*, vol. 56, pp. 39, 2004.

[20] Donachie, M. J. Titanium: A Technical Guide. Materials Park, OH: ASM International, 2004.

[21] Ravi Chandran, K. S.and Miracle D. B., "Titanium-Boron Alloys and Composites: Processing, Properties, and Applications," *JOM*, vol. 56, pp. 32, 41, 2004.

[22] Boehlert, C. J., Cowen C. J., Tamirisakandala S., McEldowney D. J.and Miracle D. B., "In situ scanning electron microscopy observations of tensile deformation in a boron-modified Ti-6Al-4V alloy," *Scripta Materialia*, vol. 55, pp. 465-468, 2006.

[23] Alman, D. E.and Hawk J. A., "The abrasive wear of sintered titanium matrix-ceramic particle reinforced composites," *Wear*, vol. 225-229, pp. 629-639, 1999.

[24] Xinghong, Z., Qiang X., Jiecai H.and Kvanin V. L., "Self-propagating high temperature combustion synthesis of TiB/Ti composites," *Materials Science and Engineering A*, vol. 348, pp. 41-46, 2003.

[25] Peng, Z., Bhaduri S. B.and Watt G. L. Synthesis of TiB-$TiB_2$ Reinforced Titanium Matrix Composites by Combustion Synthesis. In: Logan KV, Munir ZA, Spriggs RM, editors. Advanced Synthesis and Processing of Composites and Advanced Ceramics II: The American Ceramic Society, 1996. p.11-18.

[26] Kooi, B. J., Pei Y. T.and De Hosson J. T. M., "The evolution of microstructure in a laser clad TiB-Ti composite coating," *Acta Materialia*, vol. 51, pp. 831-845, 2003.

[27] Geng, K., Lu W., Qin Y.and Zhang D., "In situ preparation of titanium matrix composites reinforced with TiB whiskers and $Y_2O_3$ particles," *Materials Research Bulletin*, vol. 39, pp. 873-879, 2004.

[28] Tamirisakandala, S., Bhat R. B., Ravi V. A.and Miracle D. B., "Powder Metallurgy Ti-6Al-4V-xB Alloys: Processing, Microstructure, and Properties," *JOM*, vol. 56, pp. 60-63, 2004.

[29] Godfrey, T. M. T., Wisbey A., Goodwin P. S., Bagnall K.and Ward-Close C. M., "Microstructure and tensile properties of mechanically alloyed Ti-6A1-4V with boron additions," *Materials Science and Engineering A*, vol. 282, pp. 240-250, 2000.

[30] Kawabata, K., Sato E.and Kuribayashi K., "Composite weakening and strengthening of Ti/TiB(w) composites during steady-state creep at high temperature in the [beta]-matrix region," *Scripta Materialia*, vol. 50, pp. 523-527, 2004.

[31] Rhee, S. I., Nam S. W.and Hagiwara M., "Effect of $TiB_p$ particle reinforcement on the creep resistance of near [alpha] titanium alloy made by blended elemental powder metallurgy," *Journal of Alloys and Compounds*, vol. 359, pp. 186-192, 2003.

[32] Lieberman, S. I. Micorstructural characterization, visualization, and simulation of Ti-B materials. Materials Science and Engineering, vol. PhD. Atlanta: Georgia Institute of Technology, 2007.

[33] Schuh, C.and Dunand D. C., "Whisker alignment of Ti-6Al-4V/TiB composites during deformation by transformation superplasticity," *International Journal of Plasticity*, vol. 17, pp. 317-340, 2001.

[34] Tamirisakandala, S., Vedam B. V.and Bhat R. B., "Recent Advances in the Deformation Processing of Titanium Alloys," *Journal of Materials Engineering and Performance*, vol. 12, pp. 661-673, 2003.

[35] Massalski, T. B., Murray J. L., Bennett L. H.and Baker H., editors. Binary Alloy Phase Diagrams. Metals Park, OH: American Society for Metals, 1986.

[36] Tamirisakandala, S., Bhat R. B., Tiley J. S.and Miracle D. B., "Grain refinement of cast titanium alloys via trace boron addition," *Scripta Materialia*, vol. 53, pp. 1421-1426, 2005.

[37] Constantinides, G., Ravi Chandran K. S., Ulm F.-J.and Van Vliet K. J., "Grid indentation analysis of composite microstructure and mechanics: Principles and validation," *Materials Science and Engineering: A*, vol. 430, pp. 189-202, 2006.

[38] Gorsse, S.and Miracle D. B., "Mechanical properties of Ti-6Al-4V/TiB composites with randomly oriented and aligned TiB reinforcements," *Acta Materialia*, vol. 51, pp. 2427-2442, 2003.

[39] Gorsse, S., Petitcorps Y. L., Matar S.and Rebillat F., "Investigation of the Young's modulus of TiB needles in situ produced in titanium matrix composite," *Materials Science and Engineering A*, vol. 340, pp. 80-87, 2003.

[40] Hanusiak, W., Yolton C. F., Fields J., Hammond V.and Grabow R., "The Prospects for Hybrid Fiber-Reinforced Ti-TiB-Matrix Composites," *JOM*, vol. 56, pp. 49-50, 2004.

[41] Yolton, C. F., "The Pre-Alloyed Powder Metallurgy of Titanium with Boron and Carbon Additions," *JOM*, vol. 56, pp. 56-59, 2004.

[42] Tamirisakandala, S., Miracle D. B., Srinivasan R.and Gunasekera J. S., "Titanium Alloyed with Boron," *Advanced Materials & Processes*, vol. 162, pp. 41-43, 2006.

[43] Abkowitz, S., Abkowitz S. M., Fisher H.and Schwartz P. J., "CermeTi Discontinuously Reinforced Ti-Matrix Composites: Manufacturing, Properties, and Applications," *JOM*, vol. 56, pp. 37-41, 2004.

[44] Abkowitz, S., Abkowitz S. M., Heussi H. L.and Weihrauch P. F. Titanium composite skate blades. In: USPTO, editor. USA: Dynamet Technology, 2001.

[45] Abkowitz, S., Abkowitz S. M., Heussi H. L.and Weihrauch P. F. Titanium composite skate blades. In: USPTO, editor. US: Dynamet Technology, 2003.

[46] Zhu, J., Kamiya A., Yamada T., Shi W.and Naganuma K., "Influence of boron addition on microstructure and mechanical properties of dental cast titanium alloys," *Materials Science and Engineering A*, vol. 339, pp. 53-62, 2003.

[47] Gokhale, A. M. ASM Metals Handbook Volume 9: Metallography and Microstructures. Ohio: ASM International, 2004.

[48] Stoyan, D., Kendall S.and Mecke J. Stochastic Geometry and its Applications, second ed. New York: John Wiley and Sons, 1995.

[49] Sahimi, M. Heterogeneous Materials I: Linear Transport and Optical Properties: Springer-Verlag, 2003.

[50] Torquato, S. Random Heterogeneous Materials: Microstructure and Macroscopic Properties: Springer-Verlag, 2002.

[51] Corson, P. B., "Correlation-Functions for Predicting Properties of Heterogeneous Materials .1. Experimental Measurement of Spatial Correlation-Functions in Multiphase Solids," *Journal of Applied Physics*, vol. 45, pp. 3159-3164, 1974.

[52] Frisch, H. L.and Stillinger F. H., "Contribution to Statistical Geometric Basis of Radiation Scattering," *Journal of Chemical Physics*, vol. 38, pp. 2200-&, 1963.

[53] Tewari, A., Gokhale A. M., Spowart J. E.and Miracle D. B., "Quantitative characterization of spatial clustering in three-dimensional microstructures using two-point correlation functions," *Acta Materialia*, vol. 52, pp. 307-319, 2004.

[54] Saxl, I., "Contact Distances and Random Free Paths," *Journal of Microscopy-Oxford*, vol. 170, pp. 53-64, 1993.

[55] Mangan, M. A., Lauren P. D.and Shiflet G. J., "Three-dimensional reconstruction of Widmanstatten plates in Fe-12.3Mn-0.8C," *Journal of Microscopy-Oxford*, vol. 188, pp. 36-41, 1997.

[56] Kral, M. V., Mangan M. A., Spanos G.and Rosenberg R. O., "Three-Dimensional Analysis of Microstructures," *Materials Characterization*, vol. 45, pp. 17-23, 2000.

[57] Sidhu, R. S.and Chawla N., "Three-dimensional microstructure characterization of Ag3Sn intermetallics in Sn-rich solder by serial sectioning," *Materials Characterization*, vol. 52, pp. 225-230, 2000.

[58] Wu, K. M.and Enomoto M., "Three-Dimensional Morphology of Degenerate Ferrite in an Fe-C-Mo Alloy," *Scripta Materialia*, vol. 46, pp. 569-574, 2002.

[59] Yokomizo, T., Enomoto M., Spanos G.and Rosenberg R. O., "Three-Dimensional Distribution, Morphology, and Nucleation Sites of Intergranular Ferrite in Association With Inclusions," *Materials Science and Engineering*, vol. A344, pp. 261-267, 2003.

[60] Li, M., Ghosh S., Richmond O., Weiland H.and Rouns T. N., "Three dimensional characterization and modeling of particle reinforced metal matrix composties part II: damage characterization," *Materials Science and Engineering*, vol. A266, pp. 221-240, 1999.

[61] Tewari, A., Gokhale A. M.and German R. M., "Effect of Gravity on Three-Dimensional Coordination Number Distribution in Liquid Phase Sintered Microstructures," *Acta Materialia*, vol. 47, pp. 3721-3734, 1991.

[62] Yang, S., Tewari A.and Gokhale A. M., "Modeling of non-uniform spatial arrangement of fibers in a ceramic matrix composite," *Acta Materialia*, vol. 45, pp. 3059-3069, 1997.

[63] Tewari, A.and Gokhale A. M., "Efficient estimation of number density in opaque material microstructures: the large-area disector," *Journal of Microscopy-Oxford*, vol. 200, pp. 277-283, 2000.

[64] Tewari, A.and Gokhale A. M., "Application of Serial Sectioning for Estimation of Three-Dimensional Grain Size Distribution in a Liquid Phase Sintered Microstructure," *Materials Characterization*, vol. 46, pp. 329-335, 2001.

[65] Dighe, M. D., Tewari A., Patel G. R., Mirabelli T.and Gokhale A. M., "Application of Digital Image Processing to Reconstruct Three-Dimensional Micro-Porosity in a Cast A356.0 Alloy," *Trans. American Foundry Society*, vol. 99, pp. 353-356, 2000.

[66] Tewari, A. Ph.D. Dissertation. Georgia Institute of Technology, 1999.

[67] Spowart, J. E., Mullens H. M.and Puchala B. T., "Collecting and analyzing microstructures in three dimensions: A fully automated approach," *JOM*, vol. 55, pp. 35, 2003.

[68] Cooper, D. W., "Random-Sequential-Packing Simulations in 3 Dimensions for Spheres," *Physical Review A*, vol. 38, pp. 522-524, 1988.

[69] Louis, P.and Gokhale A. M., "Computer simulation of spatial arrangement and connectivity of particles in three-dimensional microstructure: Application to model electrical conductivity of polymer matrix composite," *Acta Materialia*, vol. 44, pp. 1519-1528, 1996.

[70] Ghosh, S., Nowak Z.and Lee K., "Quantitative characterization and modelling of composite microstructures by voronoi cells," *Acta Materialia*, vol. 45, pp. 2215-2234, 1997.

[71] Rintoul, M. D.and Torquato S., "Reconstruction of the structure of dispersions," *Journal of Colloid and Interface Science*, vol. 186, pp. 467-476, 1997.

[72] Geni, M.and Kikuchi M., "Damage analysis of aluminum matrix composite considering non-uniform distribution of SiC particles," *Acta Materialia*, vol. 46, pp. 3125-3133, 1998.

[73] Cule, D.and Torquato S., "Generating random media from limited microstructural information via stochastic optimization," *Journal of Applied Physics*, vol. 86, pp. 3428, 1999.

[74] Li, M., Ghosh S., Richmond O., Weiland H.and Rouns T. N., "Three dimensional characterization and modeling of particle reinforced metal matrix composites: Pt. I. Quantitative description of microstructural morphology," *Materials Science & Engineering A (Structural Materials: Properties, Microstructure and Processing)*, vol. A265, pp. 153-173, 1999.

[75] Manwart, C., Torquato S.and Hilfer R., "Stochastic reconstruction of sandstones," *Physical Review E*, vol. 62, pp. 893-899, 2000.

[76] Lepinoux, J.and Estrin Y., "Mechanical behaviour of alloys containing heterogeneously distributed particles: Modelling with Delaunay triangulation," *Acta Materialia*, vol. 48, pp. 4337-4347, 2000.

[77] Sheehan, N.and Torquato S., "Generating microstructures with specified correlation functions," *Journal of Applied Physics*, vol. 89, pp. 53, 2001.

[78] Shan, Z. H.and Gokhale A. M., "Digital image analysis and microstructure modeling tools for microstructure sensitive design of materials," *International Journal of Plasticity*, vol. 20, pp. 1347-1370, 2004.

[79] Tewari, A.and Gokhale A. M., "Nearest-neighbor distances between particles of finite size in three-dimensional uniform random microstructures," *Materials Science and Engineering a-Structural Materials Properties Microstructure and Processing*, vol. 385, pp. 332-341, 2004.

[80] Jefferson, G., Garmestani H., Tannenbaum R., Gokhale A. M.and Tadd E., "Two-point probability distribution fucntion analysis of co-polymer nano-composites," *International Journal of Plasticity*, vol. 21, pp. 185-198, 2005.

[81] Tscheschel, A., Lacayo J.and Stoyan D., "Statistical characterization of TEM images of silica-filled rubber," *Journal of Microscopy-Oxford*, vol. 217, pp. 75-82, 2005.

[82] Tewari, A.and Gokhale A. M., "Nearest neighbor distances in uniform-random poly-dispersed microstructures," *Materials Science and Engineering a-Structural Materials Properties Microstructure and Processing*, vol. 396, pp. 22-27, 2005.

[83] Matsumoto, M.and Nishimura T., "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, pp. 3-30, 1998.

[84] Metropolis, N., Rosenbluth A. W., Rosenbluth M. N., Teller A. H.and Teller E., "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, vol. 21, pp. 1087-1092, 1953.

[85] Serra, J., "Boolean Random Functions," *Journal of Microscopy-Oxford*, vol. 156, pp. 41-63, 1989.

[86] Yeong, C. L. Y.and Torquato S., "Reconstructing random media. II. Three-dimensional media from two-dimensiomal cuts," *Physical Review E*, vol. 58, pp. 224-233, 1998.

[87] Adler, R. J. The geometry of random fields. New York, NY: Wiley, 1981.

[88] Everett, R. K.and Chu J. H., "Modeling of Nonuniform Composite Microstructures," *Journal of Composite Materials*, vol. 27, pp. 1128-1144, 1993.

[89] Manwart, C.and Hilfer R., "Reconstruction of random media using Monte Carlo methods," *Physical Review E*, vol. 59, pp. 5596-5599, 1999.

[90] Singh, H., Gokhale A. M., Lieberman S. I.and Tamirisakandala S., "Image based computations of lineal path probability distributions

for microstructure representation," *Materials Science and Engineering a-Structural Materials Properties Microstructure and Processing*, vol. pp. 2007.

[91] Spowart, J. E.and Miracle D. B., "The influence of reinforcement morphology on the tensile response of 6061/SiC/25p discontinuously-reinforced aluminum," *Materials Science and Engineering*, vol. A357, pp. 111-123, 2003.

[92] Spowart, J. E., Ma Z.and Mishra R. The Effect of Friction Stir Processing (FSP) on the Spatial Heterogeneity of Discontinuously-Reinforced Aluminum (DRA) Composites. In: Jatta KV, Mahony M, Mishra R, editors. Friction Stir Welding and Processing II, 2003.

[93] Lu, B.and Torquato S., "Lineal-path function for random heterogeneous materials," *Physical Review A*, vol. 45, pp. 922, 1992.

[94] Louis, P.and Gokhale A. M., "Application of image analysis for characterization of spatial arrangements of features in microstructure," *Metallurgical and Materials Transactions A: Physical Metallurgy and Materials Science*, vol. 26A, pp. 1449-1456, 1995.

[95] Tewari, A.and Gokhale A. M., "Efficient estimation of number density in opaque material microstructures: the large-area disector," *Journal of Microscopy*, vol. 200, pp. 277-283, 2000.

[96] Keppel, E., "Approximating Complex Surfaces by Triangulation of Contour Lines," *IBM Journal of Research and Development*, vol. 19, pp. 2-11, 1975.

[97] Lorensen, W. E.and Cline H. E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics (ACM)*, vol. 21, pp. 163-169, 1987.

[98] Sabella, P., "Rendering algorithm for visualizing 3D scalar fields," *Computer Graphics (ACM)*, vol. 22, pp. 51-58, 1988.

[99] Caceres, C. H.and Griffiths J. R., "Damage by the cracking of silicon particles in an Al-7Si-0.4Mg casting alloy," *Acta Materialia*, vol. 44, pp. 25-33, 1996.

[100] Gurland, J.and Plateau J., "The mechanism of ductile rupture of metals containing. inclusions.," *Trans. ASM*, vol. 56, pp. 442-454, 1963.

[101] Yeh, J.-W.and Liu W.-P., "The cracking mechanism of silicon particles in an A357 aluminum alloy," *Metall & Mater Trans.-A*, vol. 27, pp. 3558-3568, 1996.

[102] Tewari, A., Gokhale A. M., Spowart J. E.and Miracle D. B., "Quantitative Characterization of Spatial Clustering in Three-Dimensional Microstructures Using Two-Point Correlation Functions," *Acta Materialia*, vol. 52, pp. 307-319, 2004.

[103] Rhines, F. N.and Craig K. R., "Measurement of Average Grain Volume and Topological Parameters by Serial Sectioning Analysis," *Metallurgical Transactions A*, vol. 7A, pp. 1729-1734, 1976.

[104] Sidhu, R. S.and Chawla N., "Three-dimensional microstructure characterization of $Ag_3Sn$ intermetallics in Sn-rich solder by serial sectioning," *Materials Characterization*, vol. 52, pp. 225-230, 2004.

[105] Shan, Z. H.and Gokhale A. M., "Micromechanics of complex three-dimensional microstructures," *Acta Materialia*, vol. 49, pp. 2001-2015, 2001.

[106] Ren, M. W., Yang J. Y.and Sun H., "Tracing boundary contours in a binary image," *Image and Vision Computing*, vol. 20, pp. 125-131, 2002.